

Introduction
to
the Internet Protocols

C R
C S
Computer Science Facilities Group
C I
L S

RUTGERS
The State University of New Jersey

3 July 1987

This is an introduction to the Internet networking protocols (TCP/IP). It includes a summary of the facilities available and brief descriptions of the major protocols in the family.

Copyright (C) 1987, Charles L. Hedrick. Anyone may reproduce this document, in whole or in part, provided that: (1) any copy or republication of the entire document must show Rutgers University as the source, and must include this notice; and (2) any other use of this material must reference this manual and Rutgers University, and the fact that the material is copyright by Charles Hedrick and is used by permission.

Unix is a trademark of AT&T Technologies, Inc.

THE UNIVERSITY OF CHICAGO
LIBRARY

THE UNIVERSITY OF CHICAGO LIBRARY

THE UNIVERSITY OF CHICAGO LIBRARY

THE UNIVERSITY OF CHICAGO LIBRARY

THE UNIVERSITY OF CHICAGO LIBRARY

THE UNIVERSITY OF CHICAGO LIBRARY

THE UNIVERSITY OF CHICAGO LIBRARY

Table of Contents

1. What is TCP/IP?	1
2. General description of the TCP/IP protocols	5
2.1 The TCP level	7
2.2 The IP level	10
2.3 The Ethernet level	11
3. Well-known sockets and the applications layer	12
3.1 An example application: SMTP	15
4. Protocols other than TCP: UDP and ICMP	17
5. Keeping track of names and information: the domain system	18
6. Routing	20
7. Details about Internet addresses: subnets and broadcasting	21
8. Datagram fragmentation and reassembly	23
9. Ethernet encapsulation: ARP	24
10. Getting more information	25

Table of Contents

1	1. Introduction
2	2. Objectives of the Study
3	3. Theoretical Framework
4	4. Methodology
5	5. Data Collection
6	6. Data Analysis
7	7. Results
8	8. Discussion
9	9. Conclusion
10	10. References
11	11. Appendix
12	12. Bibliography
13	13. Glossary
14	14. Index
15	15. Summary
16	16. Acknowledgements
17	17. Declaration
18	18. Certificate
19	19. Curriculum Vitae
20	20. List of Figures
21	21. List of Tables
22	22. List of Abbreviations
23	23. List of Symbols
24	24. List of Equations
25	25. List of References

This document is a brief introduction to TCP/IP, followed by advice on what to read for more information. This is not intended to be a complete description. It can give you a reasonable idea of the capabilities of the protocols. But if you need to know any details of the technology, you will want to read the standards yourself. Throughout the text, you will find references to the standards, in the form of "RFC" or "IEN" numbers. These are document numbers. The final section of this document tells you how to get copies of those standards.

1. What is TCP/IP?

TCP/IP is a set of protocols developed to allow cooperating computers to share resources across a network. It was developed by a community of researchers centered around the ARPANet. Certainly the ARPANet is the best-known TCP/IP network. However as of June, 87, at least 130 different vendors had products that support TCP/IP, and thousands of networks of all kinds use it.

First some basic definitions. The most accurate name for the set of protocols we are describing is the "Internet protocol suite". TCP and IP are two of the protocols in this suite. (They will be described below.) Because TCP and IP are the best known of the protocols, it has become common to use the term TCP/IP or IP/TCP to refer to the whole family. It is probably not worth fighting this habit. However this can lead to some oddities. For example, I find myself talking about NFS as being based on TCP/IP, even though it doesn't use TCP at all. (It does use IP. But it uses an alternative protocol, UDP, instead of TCP. All of this alphabet soup will be unscrambled in the following pages.)

The Internet is a collection of networks, including the Arpanet, NSFnet, regional networks such as NYsernet, local networks at a number of University and research institutions, and a number of military networks. The term "Internet" applies to this entire set of networks. The subset of them that is managed by the Department of Defense is referred to as the "DDN" (Defense Data Network). This includes some research-oriented networks, such as the Arpanet, as well as more strictly military ones. (Because much of the funding for Internet protocol developments is done via the DDN organization, the terms Internet and DDN can sometimes seem equivalent.) All of these networks are connected to each other. Users can send messages from any of them to any other, except where there are security or other policy restrictions on access. Officially speaking, the Internet protocol documents are simply standards adopted by the Internet community for its own use. More recently, the Department of Defense issued a MILSPEC definition of TCP/IP. This was intended to be a more formal definition, appropriate for use in purchasing specifications. However most of the TCP/IP community continues to use the Internet standards. The MILSPEC version is intended to be consistent with it.

Whatever it is called, TCP/IP is a family of protocols. A few provide

"low-level" functions needed for many applications. These include IP, TCP, and UDP. (These will be described in a bit more detail later.) Others are protocols for doing specific tasks, e.g. transferring files between computers, sending mail, or finding out who is logged in on another computer. Initially TCP/IP was used mostly between minicomputers or mainframes. These machines had their own disks, and generally were self-contained. Thus the most important "traditional" TCP/IP services are:

- file transfer. The file transfer protocol (FTP) allows a user on any computer to get files from another computer, or to send files to another computer. Security is handled by requiring the user to specify a user name and password for the other computer. Provisions are made for handling file transfer between machines with different character set, end of line conventions, etc. This is not quite the same thing as more recent "network file system" or "netbios" protocols, which will be described below. Rather, FTP is a utility that you run any time you want to access a file on another system. You use it to copy the file to your own system. You then work with the local copy. (See RFC 959 for specifications for FTP.)
- remote login. The network terminal protocol (TELNET) allows a user to log in on any other computer on the network. You start a remote session by specifying a computer to connect to. From that time until you finish the session, anything you type is sent to the other computer. Note that you are really still talking to your own computer. But the telnet program effectively makes your computer invisible while it is running. Every character you type is sent directly to the other system. Generally, the connection to the remote computer behaves much like a dialup connection. That is, the remote system will ask you to log in and give a password, in whatever manner it would normally ask a user who had just dialed it up. When you log off of the other computer, the telnet program exits, and you will find yourself talking to your own computer. Microcomputer implementations of telnet generally include a terminal emulator for some common type of terminal. (See RFC's 854 and 855 for specifications for telnet. By the way, the telnet protocol should not be confused with Telenet, a vendor of commercial network services.)
- computer mail. This allows you to send messages to users on other computers. Originally, people tended to use only one or two specific computers. They would maintain "mail files" on those machines. The computer mail system is simply a way for you to add a message to another user's mail file. There are some problems with this in an environment where microcomputers are used. The most serious is that a micro is not well suited to receive computer mail. When you send mail, the mail software expects to be able to open a connection to the addressee's computer, in order to send the mail. If this is a microcomputer, it may be turned off, or it may be running an application other than the mail system. For this reason, mail is normally handled by a larger system, where it is practical to have a mail server running all the time. Microcomputer mail software then becomes a

user interface that retrieves mail from the mail server. (See RFC 821 and 822 for specifications for computer mail. See RFC 937 for a protocol designed for microcomputers to use in reading mail from a mail server.)

These services should be present in any implementation of TCP/IP, except that micro-oriented implementations may not support computer mail. These traditional applications still play a very important role in TCP/IP-based networks. However more recently, the way in which networks are used has been changing. The older model of a number of large, self-sufficient computers is beginning to change. Now many installations have several kinds of computers, including microcomputers, workstations, minicomputers, and mainframes. These computers are likely to be configured to perform specialized tasks. Although people are still likely to work with one specific computer, that computer will call on other systems on the net for specialized services. This has led to the "server/client" model of network services. A server is a system that provides a specific service for the rest of the network. A client is another system that uses that service. (Note that the server and client need not be on different computers. They could be different programs running on the same computer.) Here are the kinds of servers typically present in a modern computer setup. Note that these computer services can all be provided within the framework of TCP/IP.

- network file systems. This allows a system to access files on another computer in a somewhat more closely integrated fashion than FTP. A network file system provides the illusion that disks or other devices from one system are directly connected to other systems. There is no need to use a special network utility to access a file on another system. Your computer simply thinks it has some extra disk drives. These extra "virtual" drives refer to the other system's disks. This capability is useful for several different purposes. It lets you put large disks on a few computers, but still give others access to the disk space. Aside from the obvious economic benefits, this allows people working on several computers to share common files. It makes system maintenance and backup easier, because you don't have to worry about updating and backing up copies on lots of different machines. A number of vendors now offer high-performance diskless computers. These computers have no disk drives at all. They are entirely dependent upon disks attached to common "file servers". (See RFC's 1001 and 1002 for a description of PC-oriented NetBIOS over TCP. In the workstation and minicomputer area, Sun's Network File System is more likely to be used. Protocol specifications for it are available from Sun Microsystems.)
- remote printing. This allows you to access printers on other computers as if they were directly attached to yours. (The most commonly used protocol is the remote lineprinter protocol from Berkeley Unix. Unfortunately, there is no protocol document for this. However the C code is easily obtained from Berkeley, so implementations are common.)

- remote execution. This allows you to request that a particular program be run on a different computer. This is useful when you can do most of your work on a small computer, but a few tasks require the resources of a larger system. There are a number of different kinds of remote execution. Some operate on a command by command basis. That is, you request that a specific command or set of commands should run on some specific computer. (More sophisticated versions will choose a system that happens to be free.) However there are also "remote procedure call" systems that allow a program to call a subroutine that will run on another computer. (There are many protocols of this sort. Berkeley Unix contains two servers to execute commands remotely: rsh and rexec. The man pages describe the protocols that they use. The user-contributed software with Berkeley 4.3 contains a "distributed shell" that will distribute tasks among a set of systems, depending upon load. Remote procedure call mechanisms have been a topic for research for a number of years, so many organizations have implementations of such facilities. The most widespread commercially-supported remote procedure call protocols seem to be Xerox's Courier and Sun's RPC. Protocol documents are available from Xerox and Sun. There is a public implementation of Courier over TCP as part of the user-contributed software with Berkeley 4.3. An implementation of RPC was posted to Usenet by Sun, and also appears as part of the user-contributed software with Berkeley 4.3.)
- name servers. In large installations, there are a number of different collections of names that have to be managed. This includes users and their passwords, names and network addresses for computers, and accounts. It becomes very tedious to keep this data up to date on all of the computers. Thus the databases are kept on a small number of systems. Other systems access the data over the network. (RFC 822 and 823 describe the name server protocol used to keep track of host names and Internet addresses on the Internet. This is now a required part of any TCP/IP implementation. IEN 116 describes an older name server protocol that is used by a few terminal servers and other products to look up host names. Sun's Yellow Pages system is designed as a general mechanism to handle user names, file sharing groups, and other databases commonly used by Unix systems. It is widely available commercially. Its protocol definition is available from Sun.)
- terminal servers. Many installations no longer connect terminals directly to computers. Instead they connect them to terminal servers. A terminal server is simply a small computer that only knows how to run telnet (or some other protocol to do remote login). If your terminal is connected to one of these, you simply type the name of a computer, and you are connected to it. Generally it is possible to have active connections to more than one computer at the same time. The terminal server will have provisions to switch between connections rapidly, and to notify you when output is waiting for another connection. (Terminal servers use the telnet protocol, already mentioned. However any real terminal server will also have to support name service and a

number of other protocols.)

- network-oriented window systems. Until recently, high-performance graphics programs had to execute on a computer that had a bit-mapped graphics screen directly attached to it. Network window systems allow a program to use a display on a different computer. Full-scale network window systems provide an interface that lets you distribute jobs to the systems that are best suited to handle them, but still give you a single graphically-based user interface. (The most widely-implemented window system is X. A protocol description is available from MIT's Project Athena. A reference implementation is publically available from MIT. A number of vendors are also supporting NEWS, a window system defined by Sun. Both of these systems are designed to use TCP/IP.)

Note that some of the protocols described above were designed by Berkeley, Sun, or other organizations. Thus they are not officially part of the Internet protocol suite. However they are implemented using TCP/IP, just as normal TCP/IP application protocols are. Since the protocol definitions are not considered proprietary, and since commercially-support implementations are widely available, it is reasonable to think of these protocols as being effectively part of the Internet suite. Note that the list above is simply a sample of the sort of services available through TCP/IP. However it does contain the majority of the "major" applications. The other commonly-used protocols tend to be specialized facilities for getting information of various kinds, such as who is logged in, the time of day, etc. However if you need a facility that is not listed here, we encourage you to look through the current edition of Internet Protocols (currently RFC 1011), which lists all of the available protocols, and also to look at some of the major TCP/IP implementations to see what various vendors have added.

2. General description of the TCP/IP protocols

TCP/IP is a layered set of protocols. In order to understand what this means, it is useful to look at an example. A typical situation is sending mail. First, there is a protocol for mail. This defines a set of commands which one machine sends to another, e.g. commands to specify who the sender of the message is, who it is being sent to, and then the text of the message. However this protocol assumes that there is a way to communicate reliably between the two computers. Mail, like other application protocols, simply defines a set of commands and messages to be sent. It is designed to be used together with TCP and IP. TCP is responsible for making sure that the commands get through to the other end. It keeps track of what is sent, and retransmits anything that did not get through. If any message is too large for one datagram, e.g. the text of the mail, TCP will split it up into several datagrams, and make sure that they all arrive correctly. Since these functions are needed for many applications, they are put together into a separate protocol, rather than being part

of the specifications for sending mail. You can think of TCP as forming a library of routines that applications can use when they need reliable network communications with another computer. Similarly, IP calls on the services of IP. Although the services that TCP supplies are needed by many applications, there are still some kinds of applications that don't need them. However there are some services that every application needs. So these services are put together into IP. As with TCP, you can think of IP as a library of routines that TCP calls on, but which is also available to applications that don't use TCP. This strategy of building several levels of protocol is called "layering". We think of the applications programs such as mail, TCP, and IP, as being separate "layers", each of which calls on the services of the layer below it. Generally, TCP/IP applications use 4 layers:

- an application protocol such as mail
- a protocol such as TCP that provides services need by many applications
- IP, which provides the basic service of getting datagrams to their destination
- the protocols needed to manage a specific physical medium, such as Ethernet or a point to point line.

TCP/IP is based on the "catenet model". (This is described in more detail in IEN 48.) This model assumes that there are a large number of independent networks connected together by gateways. The user should be able to access computers or other resources on any of these networks. Datagrams will often pass through a dozen different networks before getting to their final destination. The routing needed to accomplish this should be completely invisible to the user. As far as the user is concerned, all he needs to know in order to access another system is an "Internet address". This is an address that looks like 128.6.4.194. It is actually a 32-bit number. However it is normally written as 4 decimal numbers, each representing 8 bits of the address. (The term "octet" is used by Internet documentation for such 8-bit chunks. The term "byte" is not used, because TCP/IP is supported by some computers that have byte sizes other than 8 bits.) Generally the structure of the address gives you some information about how to get to the system. For example, 128.6 is a network number assigned by a central authority to Rutgers University. Rutgers uses the next octet to indicate which of the campus Ethernets is involved. 128.6.4 happens to be an Ethernet used by the Computer Science Department. The last octet allows for up to 254 systems on each Ethernet. (It is 254 because 0 and 255 are not allowed, for reasons that will be discussed later.) Note that 128.6.4.194 and 128.6.5.194 would be different systems. The structure of an Internet address is described in a bit more detail later.

Of course we normally refer to systems by name, rather than by Internet address. When we specify a name, the network software looks it up in a database, and comes up with the corresponding Internet address. Most of the network software deals strictly in terms of the

address. (RFC 882 describes the name server technology used to handle this lookup.)

TCP/IP is built on "connectionless" technology. Information is transferred as a sequence of "datagrams". A datagram is a collection of data that is sent as a single message. Each of these datagrams is sent through the network individually. There are provisions to open connections (i.e. to start a conversation that will continue for some time). However at some level, information from those connections is broken up into datagrams, and those datagrams are treated by the network as completely separate. For example, suppose you want to transfer a 15000 octet file. Most networks can't handle a 15000 octet datagram. So the protocols will break this up into something like 30 500-octet datagrams. Each of these datagrams will be sent to the other end. At that point, they will be put back together into the 15000-octet file. However while those datagrams are in transit, the network doesn't know that there is any connection between them. It is perfectly possible that datagram 14 will actually arrive before datagram 13. It is also possible that somewhere in the network, an error will occur, and some datagram won't get through at all. In that case, that datagram has to be sent again.

Note by the way that the terms "datagram" and "packet" often seem to be nearly interchangeable. Technically, datagram is the right word to use when describing TCP/IP. A datagram is a unit of data, which is what the protocols deal with. A packet is a physical thing, appearing on an Ethernet or some wire. In most cases a packet simply contains a datagram, so there is very little difference. However they can differ. When TCP/IP is used on top of X.25, the X.25 interface breaks the datagrams up into 128-byte packets. This is invisible to IP, because the packets are put back together into a single datagram at the other end before being processed by TCP/IP. So in this case, one IP datagram would be carried by several packets. However with most media, there are efficiency advantages to sending one datagram per packet, and so the distinction tends to vanish.

2.1 The TCP level

Two separate protocols are involved in handling TCP/IP datagrams. TCP (the "transmission control protocol") is responsible for breaking up the message into datagrams, reassembling them at the other end, resending anything that gets lost, and putting things back in the right order. IP (the "internet protocol") is responsible for routing individual datagrams. It may seem like TCP is doing all the work. And in small networks that is true. However in the Internet, simply getting a datagram to its destination can be a complex job. A connection may require the datagram to go through several networks at Rutgers, a serial line to the John von Neuman Supercomputer Center, a couple of Ethernets there, a series of 56Kbaud phone lines to another NSFnet site, and more Ethernets on another campus. Keeping track of the routes to all of the destinations and handling incompatibilities among different transport media turns out to be a complex job. Note

that the interface between TCP and IP is fairly simple. TCP simply hands IP a datagram with a destination. IP doesn't know how this datagram relates to any datagram before it or after it.

It may have occurred to you that something is missing here. We have talked about Internet addresses, but not about how you keep track of multiple connections to a given system. Clearly it isn't enough to get a datagram to the right destination. TCP has to know which connection this datagram is part of. This task is referred to as "demultiplexing." In fact, there are several levels of demultiplexing going on in TCP/IP. The information needed to do this demultiplexing is contained in a series of "headers". A header is simply a few extra octets tacked onto the beginning of a datagram by some protocol in order to keep track of it. It's a lot like putting a letter into an envelope and putting an address on the outside of the envelope. Except with modern networks it happens several times. It's like you put the letter into a little envelope, your secretary puts that into a somewhat bigger envelope, the campus mail center puts that envelope into a still bigger one, etc. Here is an overview of the headers that get stuck on a message that passes through a typical TCP/IP network:

We start with a single data stream, say a file you are trying to send to some other computer:

.....
TCP breaks it up into manageable chunks. (In order to do this, TCP has to know how large a datagram your network can handle. Actually, the TCP's at each end say how big a datagram they can handle, and then they pick the smallest size.)

.....
TCP puts a header at the front of each datagram. This header actually contains at least 20 octets, but the most important ones are a source and destination "port number" and a "sequence number". The port numbers are used to keep track of different conversations. Suppose 3 different people are transferring files. Your TCP might allocate port numbers 1000, 1001, and 1002 to these transfers. When you are sending a datagram, this becomes the "source" port number, since you are the source of the datagram. Of course the TCP at the other end has assigned a port number of its own for the conversation. Your TCP has to know the port number used by the other end as well. (It finds out when the connection starts, as we will explain below.) It puts this in the "destination" port field. Of course if the other end sends a datagram back to you, the source and destination port numbers will be reversed, since then it will be the source and you will be the destination. Each datagram has a sequence number. This is used so that the other end can make sure that it gets the datagrams in the right order, and that it hasn't missed any. (See the TCP specification for details.) TCP doesn't number the datagrams, but the octets. So if there are 500 octets of data in each datagram, the first datagram might be numbered 0, the second 500, the next 1000, the next 1500, etc. Finally, I will mention the Checksum. This is a number that is computed by adding up all the octets in the datagram

(more or less - see the TCP spec). The result is put in the header. TCP at the other end computes the checksum again. If they disagree, then something bad happened to the datagram in transmission, and it is thrown away. So here's what the datagram looks like now.

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                               |                               |                               |
|      Source Port              |      Destination Port      |
|                               |                               |
|                               | Sequence Number            |
|                               |                               |
|      Acknowledgment Number   |
|                               |                               |
| Data |                        |U|U|U|P|R|R|S|I|F|          |
| Offset| Reserved             |R|R|C|S|S|S|Y||          | Window
|       |                     |G|K|H|T|N|N|          |
|                               |                               |
|                               |                               |
|      Checksum                |      Urgent Pointer        |
|                               |                               |
| your data ... next 500 octets |
|                               |

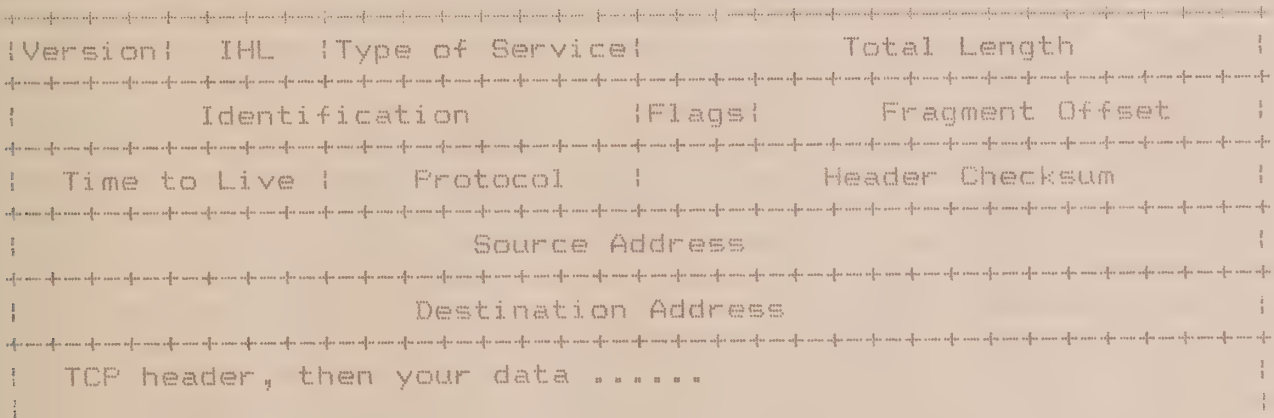
```

If we abbreviate the TCP header as "T", the whole file now looks like this:

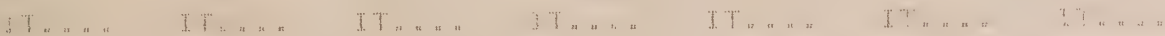
You may note that there are items in the header that I have not described above. They are generally involved with managing the connection. In order to make sure the datagram has arrived at its destination, the recipient has to send back an "acknowledgement". This is a datagram whose "Acknowledgement number" field is filled in. For example, sending a packet with an acknowledgement of 1500 indicates that you have received all the data up to octet number 1500. If the sender doesn't get an acknowledgement within a reasonable amount of time, it sends the data again. The window is used to control how much data can be in transit at any one time. It is not practical to wait for each datagram to be acknowledged before sending the next one. That would slow things down too much. On the other hand, you can't just keep sending, or a fast computer might overrun the capacity of a slow one to absorb data. Thus each end indicates how much new data it is currently prepared to absorb by putting the number of octets in its "Window" field. As the computer receives data, the amount of space left in its window decreases. When it goes to zero, the sender has to stop. As the receiver processes the data, it increases its window, indicating that it is ready to accept more data. Often the same datagram can be used to acknowledge receipt of a set of data and to give permission for additional new data (by an updated window). The "Urgent" field allows one end to tell the other to skip ahead in its processing to a particular octet. This is often useful for handling asynchronous events, for example when you type a control character or other command that interrupts output. The other fields are beyond the scope of this document.

2.2 The IP level

TCP sends each of these datagrams to IP. Of course it has to tell IP the Internet address of the computer at the other end. Note that this is all IP is concerned about. It doesn't care about what is in the datagram, or even in the TCP header. IP's job is simply to find a route for the datagram and get it to the other end. In order to allow gateways or other intermediate systems to forward the datagram, it adds its own header. The main things in this header are the source and destination Internet address (32-bit addresses, like 128.6.4.194), the protocol number, and another checksum. The source Internet address is simply the address of your machine. (This is necessary so the other end knows where the datagram came from.) The destination Internet address is the address of the other machine. (This is necessary so any gateways in the middle know where you want the datagram to go.) The protocol number tells IP at the other end to send the datagram to TCP. Although most IP traffic uses TCP, there are other protocols that can use IP, so you have to tell IP which protocol to send the datagram to. Finally, the checksum allows IP at the other end to verify that the header wasn't damaged in transit. Note that TCP and IP have separate checksums. IP needs to be able to verify that the header didn't get damaged in transit, or it could send a message to the wrong place. For reasons not worth discussing here, it is both more efficient and safer to have TCP compute a separate checksum for the TCP header and data. Once IP has tacked on its header, here's what the message looks like:



If we represent the IP header by an "I", your file now looks like this:



Again, the header contains some additional fields that have not been discussed. Most of them are beyond the scope of this document. The flags and fragment offset are used to keep track of the pieces when a datagram has to be split up. This can happen when datagrams are forwarded through a network for which they are too big. (This will be discussed a bit more below.) The time to live is a number that is decremented whenever the datagram passes through a system. When it goes to zero, the datagram is discarded. This is done in case a loop

develops in the system somehow. Of course this should be impossible, but well-designed networks are built to cope with "impossible" conditions.

At this point, it's possible that no more headers are needed. If your computer happens to have a direct phone line connecting it to the destination computer, or to a gateway, it may simply send the datagrams out on the line (though likely a synchronous protocol such as HDLC would be used, and it would add at least a few octets at the beginning and end).

2.3 The Ethernet level

However most of our networks these days use Ethernet. So now we have to describe Ethernet's headers. Unfortunately, Ethernet has its own addresses. The people who designed Ethernet wanted to make sure that no two machines would end up with the same Ethernet address. Furthermore, they didn't want the user to have to worry about assigning addresses. So each Ethernet controller comes with an address builtin from the factory. In order to make sure that they would never have to reuse addresses, the Ethernet designers allocated 48 bits for the Ethernet address. People who make Ethernet equipment have to register with a central authority, to make sure that the numbers they assign don't overlap any other manufacturer. Ethernet is a "broadcast medium". That is, it is in effect like an old party line telephone. When you send a packet out on the Ethernet, every machine on the network sees the packet. So something is needed to make sure that the right machine gets it. As you might guess, this involves the Ethernet header. Every Ethernet packet has a 14-octet header that includes the source and destination Ethernet address, and a type code. Each machine is supposed to pay attention only to packets with its own Ethernet address in the destination field. (It's perfectly possible to cheat, which is one reason that Ethernet communications are not terribly secure.) Note that there is no connection between the Ethernet address and the Internet address. Each machine has to have a table of what Ethernet address corresponds to what Internet address. (We will describe how this table is constructed a bit later.) In addition to the addresses, the header contains a type code. The type code is to allow for several different protocol families to be used on the same network. So you can use TCP/IP, DECnet, Xerox NS, etc. at the same time. Each of them will put a different value in the type field. Finally, there is a checksum. The Ethernet controller computes a checksum of the entire packet. When the other end receives the packet, it recomputes the checksum, and throws the packet away if the answer disagrees with the original. The checksum is put on the end of the packet, not in the header. The final result is that your message looks like this:


```

+++++
| Ethernet destination address (first 32 bits) |
+++++
| Ethernet dest (last 16 bits) | Ethernet source (first 16 bits) |
+++++
| Ethernet source address (last 32 bits) |
+++++
| Type code |
+++++
| IP header, then TCP header, then your data |
|
|
| end of your data |
+++++
| Ethernet Checksum |
+++++

```

If we represent the Ethernet header with "E", and the Ethernet checksum with "C", your file now looks like this:

```
EIT....C  EIT....C  EIT....C  EIT....C  EIT....C
```

When these packets are received by the other end, of course all the headers are removed. The Ethernet interface removes the Ethernet header and the checksum. It looks at the type code. Since the type code is the one assigned to IP, the Ethernet device driver passes the datagram up to IP. IP removes the IP header. It looks at the IP protocol field. Since the protocol type is TCP, it passes the datagram up to TCP. TCP now looks at the sequence number. It uses the sequence numbers and other information to combine all the datagrams into the original file.

The ends our initial summary of TCP/IP. There are still some crucial concepts we haven't gotten to, so we'll now go back and add details in several areas. (For detailed descriptions of the items discussed here see, RFC 793 for TCP, RFC 791 for IP, and RFC's 894 and 896 for sending IP over Ethernet.)

3. Well-known sockets and the applications layer

So far, we have described how a stream of data is broken up into datagrams, sent to another computer, and put back together. However something more is needed in order to accomplish anything useful. There has to be a way for you to open a connection to a specified computer, log into it, tell it what file you want, and control the transmission of the file. (If you have a different application in mind, e.g. computer mail, some analogous protocol is needed.) This is done by "application protocols". The application protocols run "on top" of TCP/IP. That is, when they want to send a message, they give the message to TCP. TCP makes sure it gets delivered to the other end. Because TCP and IP take care of all the networking details, the

applications protocols can treat a network connection as if it were a simple byte stream, like a terminal or phone line.

Before going into more details about applications programs, we have to describe how you find an application. Suppose you want to send a file to a computer whose Internet address is 128.6.4.7. To start the process, you need more than just the Internet address. You have to connect to the FTP server at the other end. In general, network programs are specialized for a specific set of tasks. Most systems have separate programs to handle file transfers, remote terminal logins, mail, etc. When you connect to 128.6.4.7, you have to specify that you want to talk to the FTP server. This is done by having "well-known sockets" for each server. Recall that TCP uses port numbers to keep track of individual conversations. User programs normally use more or less random port numbers. However specific port numbers are assigned to the programs that sit waiting for requests. For example, if you want to send a file, you will start a program called "ftp". It will open a connection using some random number, say 1234, for the port number on its end. However it will specify port number 21 for the other end. This is the official port number for the FTP server. Note that there are two different programs involved. You run ftp on your side. This is a program designed to accept commands from your terminal and pass them on to the other end. The program that you talk to on the other machine is the FTP server. It is designed to accept commands from the network connection, rather than an interactive terminal. There is no need for your program to use a well-known socket number for itself. Nobody is trying to find it. However the servers have to have well-known numbers, so that people can open connections to them and start sending them commands. The official port numbers for each program are given in "Assigned Numbers".

Note that a connection is actually described by a set of 4 numbers: the Internet address at each end, and the TCP port number at each end. Every datagram has all four of those numbers in it. (The Internet addresses are in the IP header, and the TCP port numbers are in the TCP header.) In order to keep things straight, no two connections can have the same set of numbers. However it is enough for any one number to be different. For example, it is perfectly possible for two different users on a machine to be sending files to the same other machine. This could result in connections with the following parameters:

	Internet addresses	TCP ports
connection 1	128.6.4.194, 128.6.4.7	1234, 21
connection 2	128.6.4.194, 128.6.4.7	1235, 21

Since the same machines are involved, the Internet addresses are the same. Since they are both doing file transfers, one end of the connection involves the well-known port number for FTP. The only thing that differs is the port number for the program that the users are running. That's enough of a difference. Generally, at least one end of the connection asks the network software to assign it a port number that is guaranteed to be unique. Normally, it's the user's end, since the server has to use a well-known number.

Now that we know how to open connections, let's get back to the applications programs. As mentioned earlier, once TCP has opened a connection, we have something that might as well be a simple wire. All the hard parts are handled by TCP and IP. However we still need some agreement as to what we send over this connection. In effect this is simply an agreement on what set of commands the application will understand, and the format in which they are to be sent. Generally, what is sent is a combination of commands and data. They use context to differentiate. For example, the mail protocol works like this: Your mail program opens a connection to the mail server at the other end. Your program gives it your machine's name, the sender of the message, and the recipients you want it sent to. It then sends a command saying that it is starting the message. At that point, the other end stops treating what it sees as commands, and starts accepting the message. Your end then starts sending the text of the message. At the end of the message, a special mark is sent (a dot in the first column). After that, both ends understand that your program is again sending commands. This is the simplest way to do things, and the one that most applications use.

File transfer is somewhat more complex. The file transfer protocol involves two different connections. It starts out just like mail. The user's program sends commands like "log me in as this user", "here is my password", "send me the file with this name". However once the command to send data is sent, a second connection is opened for the data itself. It would certainly be possible to send the data on the same connection, as mail does. However file transfers often take a long time. The designers of the file transfer protocol wanted to allow the user to continue issuing commands while the transfer is going on. For example, the user might make an inquiry, or he might abort the transfer. Thus the designers felt it was best to use a separate connection for the data and leave the original command connection for commands. (It is also possible to open command connections to two different computers, and tell them to send a file from one to the other. In that case, the data couldn't go over the command connection.)

Remote terminal connections use another mechanism still. For remote logins, there is just one connection. It normally sends data. When it is necessary to send a command (e.g. to set the terminal type or to change some mode), a special character is used to indicate that the next character is a command. If the user happens to type that special character as data, two of them are sent.

We are not going to describe the application protocols in detail in this document. It's better to read the RFC's yourself. However there are a couple of common conventions used by applications that will be described here. First, the common network representation: TCP/IP is intended to be usable on any computer. Unfortunately, not all computers agree on how data is represented. There are differences in character codes (ASCII vs. EBCDIC), in end of line conventions (carriage return, line feed, or a representation using counts), and in whether terminals expect characters to be sent individually or a line at a time. In order to allow computers of different kinds to communicate, each applications protocol defines a standard

representation. Note that TCP and IP do not care about the representation. TCP simply sends octets. However the programs at both ends have to agree on how the octets are to be interpreted. The RFC for each application specifies the standard representation for that application. Normally it is "net ASCII". This uses ASCII characters, with end of line denoted by a carriage return followed by a line feed. For remote login, there is also a definition of a "standard terminal", which turns out to be a half-duplex terminal with echoing happening on the local machine. Most applications also make provisions for the two computers to agree on other representations that they may find more convenient. For example, PDP-10's have 36-bit words. There is a way that two PDP-10's can agree to send a 36-bit binary file. Similarly, two systems that prefer full-duplex terminal conversations can agree on that. However each application has a standard representation, which every machine must support.

3.1 An example application: SMTP

In order to give a bit better idea what is involved in the application protocols, I'm going to show an example of SMTP, which is the mail protocol. (SMTP is "simple mail transfer protocol.") We assume that a computer called TOPAZ.RUTGERS.EDU wants to send the following message.

```
Date: Sat, 27 Jun 87 13:26:31 EDT
From: hedrick@topaz.rutgers.edu
To: levy@red.rutgers.edu
Subject: meeting
```

Let's get together Monday at 1pm.

First, note that the format of the message itself is described by an Internet standard (RFC 822). The standard specifies the fact that the message must be transmitted as net ASCII (i.e. it must be ASCII, with carriage return/linefeed to delimit lines). It also describes the general structure, as a group of header lines, then a blank line, and then the body of the message. Finally, it describes the syntax of the header lines in detail. Generally they consist of a keyword and then a value.

Note that the addressee is indicated as LEVY@RED.RUTGERS.EDU. Initially, addresses were simply "person at machine". However recent standards have made things more flexible. There are now provisions for systems to handle other systems' mail. This can allow automatic forwarding on behalf of computers not connected to the Internet. It can be used to direct mail for a number of systems to one central mail server. Indeed there is no requirement that an actual computer by the name of RED.RUTGERS.EDU even exist. The name servers could be set up so that you mail to department names, and each department's mail is routed automatically to an appropriate computer. It is also possible that the part before the @ is something other than a user name. It is possible for programs to be set up to process mail. There are also provisions to handle mailing lists, and generic names such as

"postmaster" or "operator".

The way the message is to be sent to another system is described by ports 821 and 974. The program that is going to be doing the sending asks the name server several queries to determine where to route the message. The first query is to find out which machines handle mail for the name RED.RUTGERS.EDU. In this case, the server replies that RED.RUTGERS.EDU handles its own mail. The program then asks for the address of RED.RUTGERS.EDU, which is 128.6.4.2. Then the mail program opens a TCP connection to port 25 on 128.6.4.2. Port 25 is the well-known socket used for receiving mail. Once this connection is established, the mail program starts sending commands. Here is a typical conversation. Each line is labelled as to whether it is from TOPAZ or RED. Note that TOPAZ initiated the connection:

```
RED      220 RED.RUTGERS.EDU SMTP Service at 29 Jun 87 05:17:18 EDT
TOPAZ    HELO topaz.rutgers.edu
RED      250 RED.RUTGERS.EDU - Hello, TOPAZ.RUTGERS.EDU
TOPAZ    MAIL From:<hedrick@topaz.rutgers.edu>
RED      250 MAIL accepted
TOPAZ    RCPT To:<levy@red.rutgers.edu>
RED      250 Recipient accepted
TOPAZ    DATA
RED      354 Start mail input; end with <CRLF>.<CRLF>
TOPAZ    Date: Sat, 27 Jun 87 13:26:31 EDT
TOPAZ    From: hedrick@topaz.rutgers.edu
TOPAZ    To: levy@red.rutgers.edu
TOPAZ    Subject: meeting
TOPAZ
TOPAZ    Let's get together Monday at 1pm.
TOPAZ    .
RED      250 OK
TOPAZ    QUIT
RED      221 RED.RUTGERS.EDU Service closing transmission channel
```

First, note that commands all use normal text. This is typical of the Internet standards. Many of the protocols use standard ASCII commands. This makes it easy to watch what is going on and to diagnose problems. For example, the mail program keeps a log of each conversation. If something goes wrong, the log file can simply be mailed to the postmaster. Since it is normal text, he can see what was going on. It also allows a human to interact directly with the mail server, for testing. (Some newer protocols are complex enough that this is not practical. The commands would have to have a syntax that would require a significant parser. Thus there is a tendency for newer protocols to use binary formats. Generally they are structured like C or Pascal record structures.) Second, note that the responses all begin with numbers. This is also typical of Internet protocols. The allowable responses are defined in the protocol. The numbers allow the user program to respond unambiguously. The rest of the response is text, which is normally for use by any human who may be watching or looking at a log. It has no effect on the operation of the programs. (However there is one point at which the protocol uses part of the text of the response.) The commands themselves simply allow the mail program on one end to tell the mail server the

information it needs to know in order to deliver the message. In this case, the mail server could get the information by looking at the message itself. But for more complex cases, that would not be safe. Every session must begin with a HELO, which gives the name of the system that initiated the connection. Then the sender and recipients are specified. (There can be more than one RCPT command, if there are several recipients.) Finally the data itself is sent. Note that the text of the message is terminated by a line containing just a period. (If such a line appears in the message, the period is doubled.) After the message is accepted, the sender can send another message, or terminate the session as in the example above.

Generally, there is a pattern to the response numbers. The protocol defines the specific set of responses that can be sent as answers to any given command. However programs that don't want to analyze them in detail can just look at the first digit. In general, responses that begin with a 2 indicate success. Those that begin with 3 indicate that some further action is needed, as shown above. 4 and 5 indicate errors. 4 is a "temporary" error, such as a disk filling. The message should be saved, and tried again later. 5 is a permanent error, such as a non-existent recipient. The message should be returned to the sender with an error message.

(For more details about the protocols mentioned in this section, see RFC's 821/822 for mail, RFC 959 for file transfer, and RFC's 854/855 for remote logins. For the well-known port numbers, see the current edition of Assigned Numbers, and possibly RFC 814.)

4. Protocols other than TCP: UDP and ICMP

So far, we have described only connections that use TCP. Recall that TCP is responsible for breaking up messages into datagrams, and reassembling them properly. However in many applications, we have messages that will always fit in a single datagram. An example is name lookup. When a user attempts to make a connection to another system, he will generally specify the system by name, rather than Internet address. His system has to translate that name to an address before it can do anything. Generally, only a few systems have the database used to translate names to addresses. So the user's system will want to send a query to one of the systems that has the database. This query is going to be very short. It will certainly fit in one datagram. So will the answer. Thus it seems silly to use TCP. Of course TCP does more than just break things up into datagrams. It also makes sure that the data arrives, resending datagrams where necessary. But for a question that fits in a single datagram, we don't need all the complexity of TCP to do this. If we don't get an answer after a few seconds, we can just ask again. For applications like this, there are alternatives to TCP.

The most common alternative is UDP ("user datagram protocol"). UDP is designed for applications where you don't need to put sequences of datagrams together. It fits into the system much like TCP. There is

a UDP header. The network software puts the UDP header on the front of your data, just as it would put a TCP header on the front of your data. Then UDP sends the data to IP, which adds the IP header, putting UDP's protocol number in the protocol field instead of TCP's protocol number. However UDP doesn't do as much as TCP does. It doesn't split data into multiple datagrams. It doesn't keep track of what it has sent so it can resend if necessary. About all that UDP provides is port numbers, so that several programs can use UDP at once. UDP port numbers are used just like TCP port numbers. There are well-known port numbers for servers that use UDP. Note that the UDP header is shorter than a TCP header. It still has source and destination port numbers, and a checksum, but that's about it. No sequence number, since it is not needed. UDP is used by the protocols that handle name lookups (see IEN 116, RFC 882, and RFC 883), and a number of similar protocols.

Another alternative protocol is ICMP ("Internet control message protocol"). ICMP is used for error messages, and other messages intended for the TCP/IP software itself, rather than any particular user program. For example, if you attempt to connect to a host, your system may get back an ICMP message saying "host unreachable". ICMP can also be used to find out some information about the network. See RFC 792 for details of ICMP. ICMP is similar to UDP, in that it handles messages that fit in one datagram. However it is even simpler than UDP. It doesn't even have port numbers in its header. Since all ICMP messages are interpreted by the network software itself, no port numbers are needed to say where a ICMP message is supposed to go.

5. Keeping track of names and information: the domain system

As we indicated earlier, the network software generally needs a 32-bit Internet address in order to open a connection or send a datagram. However users prefer to deal with computer names rather than numbers. Thus there is a database that allows the software to look up a name and find the corresponding number. When the Internet was small, this was easy. Each system would have a file that listed all of the other systems, giving both their name and number. There are now too many computers for this approach to be practical. Thus these files have been replaced by a set of name servers that keep track of host names and the corresponding Internet addresses. (In fact these servers are somewhat more general than that. This is just one kind of information stored in the domain system.) Note that a set of interlocking servers are used, rather than a single central one. There are now so many different institutions connected to the Internet that it would be impractical for them to notify a central authority whenever they installed or moved a computer. Thus naming authority is delegated to individual institutions. The name servers form a tree, corresponding to institutional structure. The names themselves follow a similar structure. A typical example is the name BORAX.LCS.MIT.EDU. This is a computer at the Laboratory for Computer Science (LCS) at MIT. In order to find its Internet address, you might potentially have to consult 4 different servers. First, you would ask a central server

(called the root) where the EDU server is. EDU is a server that keeps track of educational institutions. The root server would give you the names and Internet addresses of several servers for EDU. (There are several servers at each level, to allow for the possibility that one might be down.) You would then ask EDU where the server for MIT is. Again, it would give you names and Internet addresses of several servers for MIT. Generally, not all of those servers would be at MIT, to allow for the possibility of a general power failure at MIT. Then you would ask MIT where the server for LCS is, and finally you would ask one of the LCS servers about BORAX. The final result would be the Internet address for BORAX.LCS.MIT.EDU. Each of these levels is referred to as a "domain". The entire name, BORAX.LCS.MIT.EDU, is called a "domain name". (So are the names of the higher-level domains, such as LCS.MIT.EDU, MIT.EDU, and EDU.)

Fortunately, you don't really have to go through all of this most of the time. First of all, the root name servers also happen to be the name servers for the top-level domains such as EDU. Thus a single query to a root server will get you to MIT. Second, software generally remembers answers that it got before. So once we look up a name at LCS.MIT.EDU, our software remembers where to find servers for LCS.MIT.EDU, MIT.EDU, and EDU. It also remembers the translation of BORAX.LCS.MIT.EDU. Each of these pieces of information has a "time to live" associated with it. Typically this is a few days. After that, the information expires and has to be looked up again. This allows institutions to change things.

The domain system is not limited to finding out Internet addresses. Each domain name is a node in a database. The node can have records that define a number of different properties. Examples are Internet address, computer type, and a list of services provided by a computer. A program can ask for a specific piece of information, or all information about a given name. It is possible for a node in the database to be marked as an "alias" (or nickname) for another node. It is also possible to use the domain system to store information about users, mailing lists, or other objects.

There is an Internet standard defining the operation of these databases, as well as the protocols used to make queries of them. Every network utility has to be able to make such queries, since this is now the official way to evaluate host names. Generally utilities will talk to a server on their own system. This server will take care of contacting the other servers for them. This keeps down the amount of code that has to be in each application program.

The domain system is particularly important for handling computer mail. There are entry types to define what computer handles mail for a given name, to specify where an individual is to receive mail, and to define mailing lists.

(See RFC's 882, 883, and 973 for specifications of the domain system. RFC 974 defines the use of the domain system in sending mail.)

6. Routing

The description above indicated that the IP implementation is responsible for getting datagrams to the destination indicated by the destination address, but little was said about how this would be done. The task of finding how to get a datagram to its destination is referred to as "routing". In fact many of the details depend upon the particular implementation. However some general things can be said.

First, it is necessary to understand the model on which IP is based. IP assumes that a system is attached to some local network. We assume that the system can send datagrams to any other system on its own network. (In the case of Ethernet, it simply finds the Ethernet address of the destination system, and puts the datagram out on the Ethernet.) The problem comes when a system is asked to send a datagram to a system on a different network. This problem is handled by gateways. A gateway is a system that connects a network with one or more other networks. Gateways are often normal computers that happen to have more than one network interface. For example, we have a Unix machine that has two different Ethernet interfaces. Thus it is connected to networks 128.6.4 and 128.6.3. This machine can act as a gateway between those two networks. The software on that machine must be set up so that it will forward datagrams from one network to the other. That is, if a machine on network 128.6.4 sends a datagram to the gateway, and the datagram is addressed to a machine on network 128.6.3, the gateway will forward the datagram to the destination. Major communications centers often have gateways that connect a number of different networks. (In many cases, special-purpose gateway systems provide better performance or reliability than general-purpose systems acting as gateways. A number of vendors sell such systems.)

Routing in IP is based entirely upon the network number of the destination address. Each computer has a table of network numbers. For each network number, a gateway is listed. This is the gateway to be used to get to that network. Note that the gateway doesn't have to connect directly to the network. It just has to be the best place to go to get there. For example at Rutgers, our interface to NSFnet is at the John von Neuman Supercomputer Center (JvNC). Our connection to JvNC is via a high-speed serial line connected to a gateway whose address is 128.6.3.12. Systems on net 128.6.3 will list 128.6.3.12 as the gateway for many off-campus networks. However systems on net 128.6.4 will list 128.6.4.1 as the gateway to those same off-campus networks. 128.6.4.1 is the gateway between networks 128.6.4 and 128.6.3, so it is the first step in getting to JvNC.

When a computer wants to send a datagram, it first checks to see if the destination address is on the system's own local network. If so, the datagram can be sent directly. Otherwise, the system expects to find an entry for the network that the destination address is on. The datagram is sent to the gateway listed in that entry. This table can get quite big. For example, the Internet now includes several hundred individual networks. Thus various strategies have been developed to reduce the size of the routing table. One strategy is to depend upon "default routes". Often, there is only one gateway out of a network.

This gateway might connect a local Ethernet to a campus-wide backbone network. In that case, we don't need to have a separate entry for every network in the world. We simply define that gateway as a "default". When no specific route is found for a datagram, the datagram is sent to the default gateway. A default gateway can even be used when there are several gateways on a network. There are provisions for gateways to send a message saying "I'm not the best gateway -- use this one instead." (The message is sent via ICMP. See RFC 792.) Most network software is designed to use these messages to add entries to their routing tables. Suppose network 128.6.4 has two gateways, 128.6.4.59 and 128.6.4.1. 128.6.4.59 leads to several other internal Rutgers networks. 128.6.4.1 leads indirectly to the NSFnet. Suppose we set 128.6.4.59 as a default gateway, and have no other routing table entries. Now what happens when we need to send a datagram to MIT? MIT is network 18. Since we have no entry for network 18, the datagram will be sent to the default, 128.6.4.59. As it happens, this gateway is the wrong one. So it will forward the datagram to 128.6.4.1. But it will also send back an error saying in effect: "to get to network 18, use 128.6.4.1". Our software will then add an entry to the routing table. Any future datagrams to MIT will then go directly to 128.6.4.1. (The error message is sent using the ICMP protocol. The message type is called "ICMP redirect.")

Most IP experts recommend that individual computers should not try to keep track of the entire network. Instead, they should start with default gateways, and let the gateways tell them the routes, as just described. However this doesn't say how the gateways should find out about the routes. The gateways can't depend upon this strategy. They have to have fairly complete routing tables. For this, some sort of routing protocol is needed. A routing protocol is simply a technique for the gateways to find each other, and keep up to date about the best way to get to every network. RFC 1009 contains a review of gateway design and routing. However rip.doc is probably a better introduction to the subject. It contains some tutorial material, and a detailed description of the most commonly-used routing protocol.

7. Details about Internet addresses: subnets and broadcasting

As indicated earlier, Internet addresses are 32-bit numbers, normally written as 4 octets (in decimal), e.g. 128.6.4.7. There are actually 3 different types of address. The problem is that the address has to indicate both the network and the host within the network. It was felt that eventually there would be lots of networks. Many of them would be small, but probably 24 bits would be needed to represent all the IP networks. It was also felt that some very big networks might need 24 bits to represent all of their hosts. This would seem to lead to 48 bit addresses. But the designers really wanted to use 32 bit addresses. So they adopted a kludge. The assumption is that most of the networks will be small. So they set up three different ranges of address. Addresses beginning with 1 to 126 use only the first octet for the network number. The other three octets are available for the host number. Thus 24 bits are available for hosts. These numbers are

used for large networks. But there can only be 126 of these very big networks. The Arpanet is one, and there are a few large commercial networks. But few normal organizations get one of these "class A" addresses. For normal large organizations, "class B" addresses are used. Class B addresses use the first two octets for the network number. Thus network numbers are 128.1 through 191.254. (We avoid 0 and 255, for reasons that we see below. We also avoid addresses beginning with 127, because that is used by some systems for special purposes.) The last two octets are available for host addresses, giving 16 bits of host address. This allows for 64516 computers, which should be enough for most organizations. (It is possible to get more than one class B address, if you run out.) Finally, class C addresses use three octets, in the range 192.1.1 to 223.254.254. These allow only 254 hosts on each network, but there can be lots of these networks. Addresses above 223 are reserved for future use, as class D and E (which are currently not defined).

Many large organizations find it convenient to divide their network number into "subnets". For example, Rutgers has been assigned a class B address, 128.6. We find it convenient to use the third octet of the address to indicate which Ethernet a host is on. This division has no significance outside of Rutgers. A computer at another institution would treat all datagrams addressed to 128.6 the same way. They would not look at the third octet of the address. Thus computers outside Rutgers would not have different routes for 128.6.4 or 128.6.5. But inside Rutgers, we treat 128.6.4 and 128.6.5 as separate networks. In effect, gateways inside Rutgers have separate entries for each Rutgers subnet, whereas gateways outside Rutgers just have one entry for 128.6. Note that we could do exactly the same thing by using a separate class C address for each Ethernet. As far as Rutgers is concerned, it would be just as convenient for us to have a number of class C addresses. However using class C addresses would make things inconvenient for the rest of the world. Every institution that wanted to talk to us would have to have a separate entry for each one of our networks. If every institution did this, there would be far too many networks for any reasonable gateway to keep track of. By subdividing a class B network, we hide our internal structure from everyone else, and save them trouble. This subnet strategy requires special provisions in the network software. It is described in RFC 950.

0 and 255 have special meanings. 0 is reserved for machines that don't know their address. In certain circumstances it is possible for a machine not to know the number of the network it is on, or even its own host address. For example, 0.0.0.23 would be a machine that knew it was host number 23, but didn't know on what network.

255 is used for "broadcast". A broadcast is a message that you want every system on the network to see. Broadcasts are used in some situations where you don't know who to talk to. For example, suppose you need to look up a host name and get its Internet address. Sometimes you don't know the address of the nearest name server. In that case, you might send the request as a broadcast. There are also cases where a number of systems are interested in information. It is then less expensive to send a single broadcast than to send datagrams individually to each host that is interested in the information. In

order to send a broadcast, you use an address that is made by using your network address, with all ones in the part of the address where the host number goes. For example, if you are on network 128.6.4, you would use 128.6.4.255 for broadcasts. How this is actually implemented depends upon the medium. It is not possible to send broadcasts on the Arpanet, or on point to point lines. However it is possible on an Ethernet. If you use an Ethernet address with all its bits on (all ones), every machine on the Ethernet is supposed to look at that datagram.

Although the official broadcast address for network 128.6.4 is now 128.6.4.255, there are some other addresses that may be treated as broadcasts by certain implementations. For convenience, the standard also allows 255.255.255.255 to be used. This refers to all hosts on the local network. It is often simpler to use 255.255.255.255 instead of finding out the network number for the local network and forming a broadcast address such as 128.6.4.255. In addition, certain older implementations may use 0 instead of 255 to form the broadcast address. Such implementations would use 128.6.4.0 instead of 128.6.4.255 as the broadcast address on network 128.6.4. Finally, certain older implementations may not understand about subnets. Thus they consider the network number to be 128.6. In that case, they will assume a broadcast address of 128.6.255.255 or 128.6.0.0. Until support for broadcasts is implemented properly, it can be a somewhat dangerous feature to use.

Because 0 and 255 are used for unknown and broadcast addresses, normal hosts should never be given addresses containing 0 or 255. Addresses should never begin with 0, 127, or any number above 223. Addresses violating these rules are sometimes referred to as "Martians", because of rumors that the Central University of Mars is using network 225.

8. Datagram fragmentation and reassembly

TCP/IP is designed for use with many different kinds of network. Unfortunately, network designers do not agree about how big packets can be. Ethernet packets can be 1500 octets long. Arpanet packets have a maximum of around 1000 octets. Some very fast networks have much larger packet sizes. At first, you might think that IP should simply settle on the smallest possible size. Unfortunately, this would cause serious performance problems. When transferring large files, big packets are far more efficient than small ones. So we want to be able to use the largest packet size possible. But we also want to be able to handle networks with small limits. There are two provisions for this. First, TCP has the ability to "negotiate" about datagram size. When a TCP connection first opens, both ends can send the maximum datagram size they can handle. The smaller of these numbers is used for the rest of the connection. This allows two implementations that can handle big datagrams to use them, but also lets them talk to implementations that can't handle them. However this doesn't completely solve the problem. The most serious problem is that the two ends don't necessarily know about all of the steps in

between. For example, when sending data between Rutgers and Berkeley, it is likely that both computers will be on Ethernets. Thus they will both be prepared to handle 1500-octet datagrams. However the connection will at some point end up going over the Arpanet. It can't handle packets of that size. For this reason, there are provisions to split datagrams up into pieces. (This is referred to as "fragmentation".) The IP header contains fields indicating the a datagram has been split, and enough information to let the pieces be put back together. If a gateway connects an Ethernet to the Arpanet, it must be prepared to take 1500-octet Ethernet packets and split them into pieces that will fit on the Arpanet. Furthermore, every host implementation of TCP/IP must be prepared to accept pieces and put them back together. This is referred to as "reassembly".

TCP/IP implementations differ in the approach they take to deciding on datagram size. It is fairly common for implementations to use 576-byte datagrams whenever they can't verify that the entire path is able to handle larger packets. This rather conservative strategy is used because of the number of implementations with bugs in the code to reassemble fragments. Implementors often try to avoid ever having fragmentation occur. Different implementors take different approaches to deciding when it is safe to use large datagrams. Some use them only for the local network. Others will use them for any network on the same campus. 576 bytes is a "safe" size, which every implementation must support.

● Ethernet encapsulation: ARP

There was a brief discussion earlier about what IP datagrams look like on an Ethernet. The discussion showed the Ethernet header and checksum. However it left one hole: It didn't say how to figure out what Ethernet address to use when you want to talk to a given Internet address. In fact, there is a separate protocol for this, called ARP ("address resolution protocol"). (Note by the way that ARP is not an IP protocol. That is, the ARP datagrams do not have IP headers.) Suppose you are on system 128.6.4.194 and you want to connect to system 128.6.4.7. Your system will first verify that 128.6.4.7 is on the same network, so it can talk directly via Ethernet. Then it will look up 128.6.4.7 in its ARP table, to see if it already knows the Ethernet address. If so, it will stick on an Ethernet header, and send the packet. But suppose this system is not in the ARP table. There is no way to send the packet, because you need the Ethernet address. So it uses the ARP protocol to send an ARP request. Essentially an ARP request says "I need the Ethernet address for 128.6.4.7". Every system listens to ARP requests. When a system sees an ARP request for itself, it is required to respond. So 128.6.4.7 will see the request, and will respond with an ARP reply saying in effect "128.6.4.7 is 8:0:20:1:56:34". (Recall that Ethernet addresses are 48 bits. This is 6 octets. Ethernet addresses are conventionally shown in hex, using the punctuation shown.) Your system will save this information in its ARP table, so future packets will go directly. All systems treat the ARP table as a cache, and clear entries in it

if they have not been used in a certain period of time.

Note by the way that ARP requests must be sent as "broadcasts". There is no way that an ARP request can be sent directly to the right system. After all, the whole reason for sending an ARP request is that you don't know the Ethernet address. So an Ethernet address of all ones is used, i.e. ff:ff:ff:ff:ff:ff. By convention, every machine on the Ethernet is required to pay attention to packets with this as an address. So every system sees every ARP request. They all look to see whether the request is for their own address. If so, they respond. If not, they could just ignore it. (Some hosts will use ARP requests to update their knowledge about other hosts on the network, even if the request isn't for them.) Note that packets whose IP address indicates broadcast (e.g. 255.255.255.255 or 128.6.4.255) are also sent with an Ethernet address that is all ones.

10. Getting more information

This directory contains documents describing the major protocols. There are literally hundreds of documents, so we have chosen the ones that seem most important. Internet standards are called RFC's. RFC stands for Request for Comment. A proposed standard is initially issued as a proposal, and given an RFC number. When it is finally accepted, it is added to Official Internet Protocols, but it is still referred to by the RFC number. We have also included two IEN's. IEN's used to be a separate classification for more informal documents. This classification no longer exists -- RFC's are now used for all official Internet documents, and a mailing list is used for more informal reports.) The convention is that whenever an RFC is revised, the revised version gets a new number. This is fine for most purposes, but it causes problems with two documents: Assigned Numbers and Official Internet Protocols. These documents are being revised all the time, so the RFC number keeps changing. You will have to look in rfc-index.txt to find the number of the latest edition. Anyone who is seriously interested in TCP/IP should read the RFC describing IP (791). RFC 1009 is also useful. It is a specification for gateways to be used by NSFnet. As such, it contains an overview of a lot of the TCP/IP technology. You should probably also read the description of at least one of the application protocols, just to get a feel for the way things work. Mail is probably a good one (821/822). TCP (793) is of course a very basic specification. However the spec is fairly complex, so you should only read this when you have the time and patience to think about it carefully. Fortunately, the author of the major RFC's (Jon Postel) is a very good writer. The TCP RFC is far easier to read than you would expect, given the complexity of what it is describing. You can look at the other RFC's as you become curious about their subject matter.

Here is a list of the documents you are more likely to want:

rfc-index list of all RFC's

rfc1012 somewhat fuller list of all RFC's

rfc1011 Official Protocols. It's useful to scan this to see what tasks protocols have been built for. This defines which RFC's are actual standards, as opposed to requests for comments.

rfc1010 Assigned Numbers. If you are working with TCP/IP, you will probably want a hardcopy of this as a reference. It's not very exciting to read. It lists all the officially defined well-known ports and lots of other things.

rfc1009 NSFnet gateway specifications. A good overview of IP routing and gateway technology.

rfc1001/2 netBIOS: networking for PC's

rfc973 update on domains

rfc959 FTP (file transfer)

rfc950 subnets

rfc937 POP2: protocol for reading mail on PC's

rfc894 how IP is to be put on Ethernet, see also rfc825

rfc882/3 domains (the database used to go from host names to Internet address and back -- also used to handle UUCP these days). See also rfc973

rfc854/5 telnet - protocol for remote logins

rfc826 ARP - protocol for finding out Ethernet addresses

rfc821/2 mail

rfc814 names and ports - general concepts behind well-known ports

rfc793 TCP

rfc792 ICMP

rfc791 IP

rfc768 UDP

rip.doc details of the most commonly-used routing protocol

ien-116 old name server (still needed by several kinds of system)

ien-48 the Catenet model, general description of the

philosophy behind TCP/IP

The following documents are somewhat more specialized.

rfc813 window and acknowledgement strategies in TCP
rfc815 datagram reassembly techniques
rfc816 fault isolation and resolution techniques
rfc817 modularity and efficiency in implementation
rfc879 the maximum segment size option in TCP
rfc896 congestion control
rfc827,888,904,975,985
 EGP and related issues

To those of you who may be reading this document remotely instead of at Rutgers: The most important RFC's have been collected into a three-volume set, the DDN Protocol Handbook. It is available from the DDN Network Information Center, SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025 (telephone: 800-235-3155). You should be able to get them via anonymous FTP from sri-nic.arpa. File names are:

RFC's:

rfc:rfc-index.txt
rfc:rfcxxx.txt

IEN's:

ien:ien-index.txt
ien:ien-xxx.txt

rip.doc is available by anonymous FTP from topaz.rutgers.edu, as /pub/tcp-ip-docs/rip.doc.

Sites with access to UUCP but not FTP may be able to retrieve them via UUCP from UUCP host rutgers. The file names would be

RFC's:

/topaz/pub/pub/tcp-ip-docs/rfc-index.txt
/topaz/pub/pub/tcp-ip-docs/rfcxxx.txt

IEN's:

/topaz/pub/pub/tcp-ip-docs/ien-index.txt
/topaz/pub/pub/tcp-ip-docs/ien-xxx.txt
/topaz/pub/pub/tcp-ip-docs/rip.doc

Note that SRI-NIC has the entire set of RFC's and IEN's, but rutgers and topaz have only those specifically mentioned above.

P R O C O M M

Version 2.4.1

R E F E R E N C E M A N U A L

Datastorm Technologies, Inc.
P.O. Box 1471
Columbia, MO 65205
BBS: (314) 449-9401

Copyright (c) 1986 Datastorm Technologies, Inc.
All Rights Reserved

Datastorm Technologies, Inc. was previously known as PIL Software Systems.

This manual was completely updated to reflect the current state of the program as of release 2.4.

Specifications subject to change without notice.

ProComm (TM) software copyright (C) 1985, 1986 Datastorm Technologies, Inc. All rights reserved.

This document copyright (C) 1986 Datastorm Technologies, Inc. All rights reserved.

ProComm is a trademark of Datastorm Technologies, Inc.

Most of the hardware names in this manual are trademarks or trade names of specific manufacturers.

Printed in the United States of America

LICENSE

All versions of ProComm, including version 2.4, are not public domain software, nor are they free software.

ProComm is copyright (C) 1985, 1986 by Datastorm Technologies, Inc..

Non-registered users are granted a limited license to use ProComm on a trial basis for the purpose of determining whether ProComm is suitable for their needs. Use of ProComm, except for this limited purpose, requires registration. Use of non-registered copies of ProComm by any person, business, corporation, governmental agency or other entity institution is strictly forbidden.

Registration permits a user the license to use ProComm only on a single computer; a registered user may use the program on a different computer, but may not use the program on more than one computer at the same time.

No user may modify ProComm in any way, including but not limited to decompiling, disassembling or otherwise reverse engineering the program.

All users are granted a limited license to copy ProComm only for the trial use of others subject to the above limitations, and also the following:

- ProComm must be copied in unmodified form, complete with the file containing this license information.

- The full ProComm documentation must be included with the copy.

- No fee, charge or other compensation may be accepted or requested by any licensee.

- ProComm may not be distributed in conjunction with any other product.

Operators of electronic bulletin board systems (Sysops) may post ProComm for downloading by their users only as long as the above conditions are met.

Distributors of public domain or user supported software may distribute copies of ProComm subject to the above conditions only after obtaining written permission from Datastorm Technologies, Inc.. Such permission usually is granted; please write for details.

See the Ordering section for more information on registration, corporate licensing and similar topics.

WARRANTY

Datastorm Technologies, Inc. makes no warranty of any kind, express or implied, including without limitation, any warranties of merchantability and/or fitness for a particular purpose. Datastorm Technologies, Inc. shall not be liable for any damages, whether direct, indirect, special or consequential arising from a failure of this program to operate in the manner desired by the user. Datastorm Technologies, Inc. shall not be liable for any damage to data or property which may be caused directly or indirectly by use of the program.

IN NO EVENT WILL Datastorm Technologies, Inc. BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF YOUR USE OR INABILITY TO USE THE PROGRAM, OR FOR ANY CLAIM BY ANY OTHER PARTY.

ORDERING INFORMATION

A ProComm registration licenses you to use the product on a regular basis. Registration includes mailed notification of updates and priority support on our support BBS. Registered users will also be granted access to a registered user only BBS to become available sometime in late 1986. Users need register only one version of ProComm; registration includes licensed use of all upgrades.

Individual registrations for ProComm come in three forms. The first, registration only, costs \$25 and assumes you have already received a copy of the program from whatever source. We do not provide diskettes or manuals. The registration fee covers the use of the program. For \$35 dollars we offer a combination registration/diskette package. The diskette includes the latest version of the program, its documentation and some supporting programs and files. We also offer The Works!, which adds a printed, bound manual to the registration and diskette. Copies of The Works! are available for \$50.

In addition, evaluation disks are available at any time for \$10. These disks do not include registration. The fee covers diskette, postage and handling. You can also send us a formatted disk, along with a postage-paid, self-addressed return mailer to receive a copy.

Please use the enclosed order form when placing an order.

ORDERS OUTSIDE THE US: Please use your Mastercard or VISA when ordering, or send checks drawn on US banks in US dollars. We can accept non-US currency; however, you must include an additional \$5 to cover conversion and collection costs. Please include an additional \$5 to cover postage on orders of The Works! delivered outside of the US.

Order Form

Total \$

vi

Corporate and Quantity Purchases

All corporate, business, government or other commercial users of ProComm must be registered. We offer quantity discounts starting at the eleventh copy. Corporate or site licensing is also available.

For single unit orders, use the enclosed order form. We accept purchase orders in amounts over \$100 only. All other orders should be prepaid.

Orders in quantities of less than 75 units are handled as bulk purchases. We can provide either the registration/disk combo or The Works!.

Purchases of over 75 units may be handled as quantity purchases or as corporate licensing agreements. Licensing agreements allow duplication and distribution of specific numbers of copies within the licensed institution. Duplication of multiple copies is not allowed except through execution of a licensing agreement. Please write or call for details.

The quantity purchase discounts are as follows:

0- 10 copies:	no discount
11- 25 copies:	15% discount
26- 50 copies:	20% discount
51-100 copies:	25% discount
101-200 copies:	30% discount
201-300 copies:	35% discount
301+ copies:	40% discount

ALL PRICES AND DISCOUNTS ARE SUBJECT TO CHANGE WITHOUT NOTICE. Discounts are not cumulative; they apply to single orders of like products only. Unit prices are the same as for individual users.

WARNING: YOU MAY NOT USE PROCOMM WITHIN YOUR ORGANIZATION WITHOUT A PRIOR PURCHASE OR LICENSE ARRANGEMENT.

PREFACE TO VERSION 2.4

Quite a number of things have changed for this release, most notably the command language. See the accompanying file PROCM24.NEW for a complete list. Enough has changed that it might be wise to delete your existing .PRM files and create a new one. This may not be necessary in all cases; however, if strange things start to occur try that before calling in with any complaints.

Speaking of calling in: our apologies to everyone who has tried to call the BBS and couldn't get through. We are receiving a tremendous volume of calls and the phone is always busy. Please do not call the operator to complain; there are no technical problems at present, just lots of callers. Late this year we will be adding a second line for registered users only who will receive priority support.

And about support: while we try to answer all the mail, electronic and otherwise, that we get, it sometimes takes us a while to get around to it. Especially stuff from non-registered users. Please be patient, and remember that we do not guarantee to provide support of any kind to non-registered users.

About this manual: as the name implies, this is a reference manual describing the operation of ProComm. It is not intended as a tutorial on communications in general. For more general and introductory information on communications see:

Glossbrenner, Alfred. The Complete Handbook of Personal Computer Communications, 2nd edition. New York: St. Martins Press, 1985.

Jordan, Larry E. Communications and Networking for the IBM PC, 1st edition. Bowie, Maryland: Robert J. Brady Company.

WARNING: Version 2.4, like some previous versions, uses an overlay structure. This means that not all of the program is loaded into memory at once. As different sections of the program are required, they are read off the disk and into memory. All of the overlays are kept in the executable file PROCOMM.EXE.

The practical implication of the overlay scheme is that PROCOMM.EXE must always be available to read overlays from. On a floppy system, this means that you cannot remove the disk on which PROCOMM.EXE resides from the drive on which it was executed. If you do, the overlay linker assumes that the PROCOMM.EXE file is open and residing on the disk, and writes an updated file directory onto the disk, thereby overwriting any and everything on the diskette.

Obviously this is not a problem for hard disks. The solution for floppy systems is simply not to remove the PROCOMM.EXE disk from the drive. You can easily change the logged drive (via the Alt-B or CHDIR commands) to access files on other drives.

C O N T E N T S

LICENSE	iii
WARRANTY	iv
ORDERING INFORMATION	v
Order Form	vi
Corporate and Quantity Purchases	vii
PREFACE TO VERSION 2.4	viii
1. I N T R O D U C T I O N	101
Hardware Requirements	101
ProComm Files	102
The ProComm Environment Variable	102
2. G E T T I N G S T A R T E D	201
Terminal Mode	202
Terminal Emulations	203
Line Settings	203
String Translation	204
Help Screen	205
Exiting ProComm	205
3. T H E S E T U P S C R E E N	301
Modem SetUp	302
Terminal SetUp	305
Kermit SetUp	308
General SetUp	308
Host Mode SetUp	312
ASCII Transfer SetUp	314
4. M A J O R F U N C T I O N S	401
Dialing Directory	401
Automatic Redial	401
Keyboard Macros	401
Line Settings	402
Translate Table	403
Editor	403
Exit	404
Host Mode	404
Chat Mode	404
DOS Gateway	404
Command Files	404
Redisplay	405
Program Information	405
SetUp Screen	405
Kermit Server Command	405
Change Directory	405
Clear Screen	406

Toggle Duplex	406
Hang Up Phone	406
Elapsed Time	406
Print On/Off	406
Set Colors	406
Auto Answer	407
Toggle CR - CR/LF	407
Break	407
Send Files (Upload)	408
Receive Files (Download)	408
Directory	408
View a File	408
Screen Dump	408
Log Toggle / Log Hold	408
 5. D I A L I N G D I R E C T O R Y	 501
Searching for an Entry	502
Revising the Dialing Directory	502
Adding or Revising an Entry	502
Revising the Modem Command	503
Adding or Revising Long Distance Codes	503
Deleting Entries	503
Making a Call	504
Manual Dialing	504
Printing the Directory	504
Automatic Redial with Circular Dialing Queue	504
 6. F I L E T R A N S F E R	 601
Uploading Files	601
Downloading Files	601
File Transfer Protocols	602
ASCII	602
XMODEM	603
MODEM7	603
YMODEM	603
Telink	603
Kermit	603
CompuServe B File Transfers	604
WXMODEM File Transfers	604
 7. C O M M A N D F I L E S	 701
Command File Syntax	702
Top Level Commands	704
Set Commands	721
Set ASCII Commands	722
Set Kermit Commands	723
ERROR MESSAGES	724
 8. H O S T M O D E	 801
Host Mode Setup	801
Modem Setup	801
Operating System Setup	801
ProComm Setup	802
 APPENDIX A - TERMINAL EMULATION	 901
Overview	901
Digital Equipment Corporation VT-100 and VT-102	902
Mapping of VT-100 Keypad	903

Keypad Application Mode for VAX/VMS EDT Editor	903
IBM 3101	904
Televideo 900 Series	905
Digital Equipment Corporation VT-52	906
Lear Siegler ADM 3/5	907
Heath/Zenith 19	908
ADDS Viewpoint	909
WYSE 100	910
ANSI-BBS	911
APPENDIX B - COMMAND REFERENCE GUIDE	1001
APPENDIX C - ANSWERS TO COMMONLY ASKED QUESTIONS	1101
APPENDIX D - PROCOMM TECHNICAL SPECIFICATIONS	1201
APPENDIX E - USER SUPPORTED SOFTWARE	1301
APPENDIX F - PRODUCT SUPPORT	1401
I N D E X	1501

1. INTRODUCTION

ProComm is a general purpose program designed to provide easy and convenient access to a broad variety of telecommunications tasks. Most of the program is written in the "C" programming language, with some assembly language routines for optimum performance. Included in its abilities are the sorts of features one would expect to find in highly sophisticated telecommunications software:

- * the ability to emulate a number of popular terminals;
- * a dialing directory containing one hundred entries;
- * automatic redial facilities for connecting with hard to reach numbers;
- * several popular file transfer protocols including XMODEM, Kermit, Telink and more;
- * command files to control automatic logon and unattended operation;
- * a DOS gateway which allows you to execute DOS commands or other programs while you are still on line;
- * a host of additional features, including keyboard macros, disk and printer logging and many others covered in detail on the following pages.

Hardware Requirements

ProComm requires a minimum of 130K of available RAM to execute properly--that is, 130K in addition to the operating system and any resident programs, including such memory resident programs as SideKick, SuperKey, etc. If you are operating with less than 192K total RAM, ProComm might not be able to load. It runs under MS-DOS on the IBM PC, XT, AT or any close compatible, and may be used with color, composite or monochrome displays.

Because ProComm may have a relatively large number of files open at once, you must assure that the FILES parameter in your CONFIG.SYS file is large enough to accommodate them. We suggest using FILES=20 (or larger). Your DOS manual can give you more information on the CONFIG.SYS file, or see the Host Mode section of this manual.

In addition, of course, you must have a working modem. The default settings in ProComm are all established for Hayes compatible modems. If your modem is not fully Hayes compatible, consult your modem owner's guide for details on switch settings, commands, and so forth. In order for ProComm to work correctly, particularly in Host Mode, your modem's Carrier Detect (CD) must be set to follow the true state of the carrier,

not forced true (or "high") by dip-switch settings. Similarly, the CD should not be set to follow DTR ("Data Terminal Ready") but rather to follow the "true state" or "RS-232 Convention" (or however your owner's manual state it). And finally, DTR should not be forced high by dip-switch settings; it, too, should follow real state.

ProComm Files

When it is first loaded, ProComm creates several files which it will use on subsequent operation:

PROCOMM.PRM, the default parameter file;

PROCOMM.DIR, a dialing directory file;

PROCOMM.KEY, a keyboard macro file;

PROCOMM.XLT, the translate table file;

PROCOMM.HST, the Host Mode audit trail;

In addition, if you plan to use the Host Mode, you might want to create a file called PROCOMM.MSG, which holds the Host Mode welcome message.

The ProComm Environment Variable

ProComm can use the environment variable PROCOMM= to tell it where to look for its files. ProComm first searches the current directory for necessary files. If the files are not there, then ProComm will search through the directory pointed to by PROCOMM= (if it exists). If the files are found in neither place, and must be created, they will be created in the directory pointed to by PROCOMM=. If the environment variable is not set, files will be created by default in the current directory.

You can set the environment variable by issuing the DOS command

```
SET PROCOMM=pathname <CR>
```

where "pathname" is a fully qualified path name ending with a backslash. You may issue this command either from the command line or from a batch file. For example, if you have the command

```
SET PROCOMM=C:\COMM\ProComm\
```

in your AUTOEXEC file, ProComm will know to look for its files in the C:\COMM\ProComm subdirectory, and you can use the program from anywhere on your system and still have just one set of ProComm files. Be sure to remember to end the pathname with a backslash (\), or ProComm will get confused and have problems reading the necessary files.

You can clear the environment variable null with the command

```
SET PROCOMM= <CR>
```

Consult your DOS manual for more information on environment variables.

2. GETTING STARTED

You begin a ProComm session by issuing the command

```
ProComm [/S] [/B] [/Ffilename] [/M] [/D]
```

where /S indicates sound effects, /B indicates black and white operation, /Ffilename indicates a command file, /M indicates screen display mode, and /D indicates the presence of dual monitors. (Commands that are displayed in square brackets, e.g. [/S], are optional).

Command line switches are optional, and may appear in any order. They must, however, be separated by at least one blank space.

Including a "/S" on the command line suppresses ProComm produced sound effects. It does not, however, affect beeps (^G) coming from the remote computer or the alarm function, nor does it control the modem speaker. You can change the default sound settings after you have loaded ProComm by using the General SetUp option of the SetUp screen (Alt-S).

The "/B" option may be used to run the program in black and white. This feature is useful if you have a composite monitor attached to a color graphics display card. Specifying "/B" will tell ProComm to use only black, white and high intensity white for all displays. If the "/B" command line switch is omitted, ProComm will adjust itself for either color or monochrome operation. To make black and white colors the default, begin a ProComm session using the "/B" option, then save the current colors using the Alt-Z (set colors) facility.

Specify "/F" with a filename to execute a command file immediately after loading ProComm. For example, to execute the command file "EXAMPLE.CMD" as the first thing ProComm does after loading, enter

```
ProComm /Fexample.cmd <CR>
```

Command files are described in detail in Section 7.

The "/M" option tells ProComm to perform screen displays using BIOS calls rather than writing directly to the screen buffer. This option is useful for running ProComm under multi-tasking operating systems. The screen write method is also selectable through the SETUP (Alt-S) facility.

The "/D" option is used when you have both a color and a monochrome display connected to your computer. This option may cause erratic behavior if used in conjunction with an EGA card/monochrome monitor combination.

When the program begins it displays the ProComm logo and copyright notice. The first few times you use ProComm the program information

screen will also appear. Once you have read the information screen, press any key to continue. You can also display the program information screen by pressing Alt-I from Terminal Mode.

WARNING: Version 2.4, like some previous versions, uses an overlay structure. This means that not all of the program is loaded into memory at once. As different sections of the program are required, they are read off the disk and into memory. All of the overlays are kept in the executable file PROCOMM.EXE.

The practical implication of the overlay scheme is that PROCOMM.EXE must always be available to read overlays from. On a floppy system, this means that you cannot remove the disk on which PROCOMM.EXE resides from the drive on which it was executed. If you do, the overlay linker assumes that the PROCOMM.EXE file is open and residing on the disk, and writes an updated file directory onto the disk, thereby overwriting any and everything on the diskette.

Obviously this is not a problem for hard disks. The solution for floppy systems is simply not to remove the PROCOMM.EXE disk from the drive. You can easily change the logged drive (via the Alt-B or CHDIR commands) to access files on other drives.

Terminal Mode

After the opening display and the program initialization, you are left in Terminal Mode. Most of your communicating will take place here. The bottom line of the screen is reserved as a status line, but the rest of the screen is open for use. When you first enter Terminal Mode, the screen will be blank except for the status line at the bottom of the screen:

```
+-----+-----+-----+-----+-----+-----+-----+-----+
!ALT F10 HELP!ANSI-BBS! HDX !300 N81!LOG CLOSED!PRT OFF!CR!CR!
+-----+-----+-----+-----+-----+-----+-----+-----+
```

This line indicates the current status of several of ProComm's features:

ALT F10 HELP indicates that you can press Alt-F10 to activate the ProComm Help Screen. This section of the status line also indicates what is currently occurring. For example, if you activate the screen dump, then the words "SCREEN DUMP" will replace "ALT F10 HELP".

The next segment of the status line ("ANSI-BBS") indicates the currently activated terminal emulation.

The third status indicator is for duplex: "HDX" indicates half-duplex; "FDX" indicates full-duplex.

In the fourth block the status indicator reflects the modem line settings, in this case 300 baud, no parity, 8 data bits, and 1 stop bit.

The LOG status indicates whether you have activated the feature which allows "logging" incoming data directly to disk. If that feature is active, then the indicator will say "LOG OPEN".

ProComm allows the option of sending incoming data directly to your printer; if you select this option, then the printer status indicator will say "PRT ON"; otherwise it remains at "PRT OFF".

If you activate the feature which adds a line feed to all incoming carriage returns, the next indicator will show "CR-LF"; otherwise it remains at just "CR".

Similarly, the final item indicates the status of the CR/CR-LF output feature.

Most ProComm features are called from Terminal Mode and are executed in a window which leaves your original screen intact. Press Alt-F10 to display a help screen which lists all available commands. You may select commands either from the help screen or directly from Terminal Mode. You can return to Terminal Mode from most windows by pressing <ESC>.

Terminal Emulations

Most keyboard and screen functions will operate locally as you expect them to. What they do on the remote computer depends on which terminal emulation you are using. ProComm supports ten popular terminal configurations:

- | | | |
|------------------------|---------------------|-------------|
| - IBM 3101 | - DEC VT-100 | - DEC VT-52 |
| - Televideo 910/920 | - Televideo 925/950 | - Wyse 100 |
| - Lear Siegler ADM-3/5 | - Heath/Zenith 19 | - ANSI-BBS |
| - ADDS Viewpoint | | |

Appendix A contains a complete list of the functions supported for each terminal.

To change the emulation currently in use, select the Terminal SetUp option from the main SetUp screen (Alt-S), or use the Emulate command in a command file.

Line Settings

ProComm supports a variety of communications configurations. You may select the baud rate, number of data and stop bits, parity, and duplex. The program defaults to 300 baud operation, with 8 data bits, 1 stop bit, no parity and full duplex (echo off).

Use the Alt-P (Line Settings) command to review or change the active line control settings. After opening the Line Settings window, ProComm displays the current settings. Enter a number from 1 to 12 to change to a predefined setting. Use the numbers 13 through 19 to set up a customized configuration. Enter 20 to select COM1 as your active port, 21 to use COM2, 22 for COM3, and 23 for COM4.

Use the Save option (24) to make the new configuration your default. If you change the parameter settings without saving them to disk, they will be used only for the current session, and will return to their original settings the next time you use ProComm.

Once you have made your selections, press <ESC> to return to Terminal Mode. The new parameters are now in effect.

String Translation

There are a number of situations in which ProComm will translate strings going to or coming from the modem. The strings that are translated include the modem initialization string, the modem command, dialing directory numbers, long distance codes, keyboard macros, modem connect messages and several script commands. The translation allows you to send carriage returns, escape codes and other control characters to the modem, as well as providing a pause feature. ProComm translation characters can be specified in the General SetUp screen. Translation characters are provided for CR, ESC, CTRL-char and pause.

The default CR translation character is the exclamation point ("!"). Anytime ProComm encounters the CR translation character in one of the strings mentioned above, it replaces that character with a CR (ASCII 13). To send the character without translation use two of them together. For example, when "!" is the CR translate character, a keyboard macro set to "PASSWORD!" sends the string "PASSWORD" followed by a carriage return. A keyboard macro set to "GO AWAY!!" will be translated as "GO AWAY!". Since two of these characters in a row are translated as the true character, some other method is necessary to send two adjacent carriage returns. You can do so by using the CTRL-char translation described below. For example, if the CTRL translate character is set to the caret ("^") the string "^M^M" is translated as 2 carriage returns (because Ctrl-M is a CR).

The default ESC translation character is the vertical bar ("|"). When ProComm encounters this character in one of the translated strings, it replaces it with an ESC (ASCII 27). Again, to send the character without translation use two of them together. For example, when "|" is the ESC translation character, a keyboard macro set to "![2J" sends the VT100 command for clear screen (ESC [2 J). A keyboard macro set to "!! box !" will be translated as "| box |". Again, since two consecutive translation characters are translated as the true character, some other method is necessary to send two adjacent escapes. As with the CR translation character, you can indicate two consecutive escapes by using the CTRL-char translation described below. If, for example, the CTRL translation character is set to the caret ("^") the string "^^[[" is translated as 2 escapes (because Ctrl-[is an ESC).

The CTRL character translation is slightly different. It is used as a lead-in marker to indicate that the following character should be interpreted as a CTRL character. For example, using the default CTRL translate character "^" (the caret), the string "^C" would be interpreted as a Ctrl-C (ASCII 3). Again, two translation characters together are treated as the actual character, so "^C" would be translated as "C" (the caret character followed by a capital C).

The final translation character provided is the pause. If this character is encountered during translation, ProComm will pause for 1/2 second before doing anything else. The default translate pause character is the tilde ("~"). This translation character is significantly different from the others in that two pause characters together cause a 1 second pause, three together cause a one and one-half second pause, and so forth.

Multiple pause translation characters are not interpreted as a single character. If you wish to actually send that character (e.g. the tilde) you must change the setup for the pause translation character.

It is important to note the difference between the translation pause character and a modem pause character. The translation pause character causes a pause in characters being sent to the modem. This character is appropriate for pauses around a modem escape code, as in a hang-up string, or for allowing the modem to respond to something before continuing. On the other hand, a modem pause character, such as the comma used by Hayes, causes a pause in characters sent from the modem to the phone line. It is appropriate for such tasks as waiting for a second dial tone when dialing out through a PBX or office telephone system. The translation pause character and the modem pause character are not the same and should not be confused.

Help Screen

The Help Screen lists the command keystrokes used to execute a ProComm feature. The Help Screen is displayed when you press Alt-F10 while in the Terminal Mode. Pressing Alt-F10 produces this screen:

```

++-----++
!!                               P r o C o m m   H e l p                               !!
!+-----+!
! MAJOR FUNCTIONS                UTILITY FUNCTIONS                FILE FUNCTIONS        !
!
!Dialing Directory  Alt-D Program Info ..... Alt-I Send files ... PgUp !
!Automatic Redial.. Alt-R Setup Screen ..... Alt-S Receive files PgDn !
!Keyboard Macros .. Alt-M Kermit Server Cmd  Alt-K Directory .... Alt-F!
!Line Settings .... Alt-P Change Directory . Alt-B View a File .. Alt-V!
!Translate Table .. Alt-W Clear Screen ..... Alt-C Screen Dump .. Alt-G!
!Editor ..... Alt-A Toggle Duplex .... Alt-E Log Toggle .. Alt-F1!
!Exit ..... Alt-X Hang Up Phone .... Alt-H Log Hold .... Alt-F2!
!Host Mode ..... Alt-Q Elapsed Time ..... Alt-T
!Chat Mode ..... Alt-O Print On/Off ..... Alt-L
!DOS Gateway ..... Alt-F4 Set Colors ..... Alt-Z
!Command Files ... Alt-F5 Auto Answer ..... Alt-Y
!Redisplay ..... Alt-F6 Toggle CR-CR/LF . Alt-F3
!                               Break Key ..... Alt-F7
+-----+

```

You can select any of these features from the Help Screen or directly from Terminal Mode. The Help Screen is merely a help, not a required step for reaching the options, nor do you need to return to the Terminal Mode before selecting one of the options. Each of the features listed in the Help Screen is described in more detail in Section 4, and many are considered in still greater detail in other sections of this manual. You can return directly to Terminal Mode from the Help Screen by pressing any non-command key.

Exiting ProComm

You can exit ProComm at any time from Terminal Mode or the Help Screen by pressing Alt-X. You will be asked to confirm the decision to leave; press "Y" to exit or "N" to resume work.

3. THE SETUP SCREEN

ProComm allows you to define many of your own power-up defaults for system parameters. You can also change a setting temporarily, use it for the current session, then automatically go back to your default settings for the next session. Setup is reached via the Alt-S command. It is divided into 6 sections: Modem Setup, Terminal Setup, Kermit Setup, General Setup, Host Mode Setup and ASCII File Transfer Setup.

Press Alt-S from the Help Window or while in Terminal Mode to activate the SetUp facility. The screen will clear and present you with these options:

- 1) MODEM SETUP
- 2) TERMINAL SETUP
- 3) KERMIT SETUP
- 4) GENERAL SETUP
- 5) HOST MODE SETUP
- 6) ASCII TRANSFER SETUP
- 8) SAVE SETUP TO DISK

You can exit the main setup screen at any time by pressing <ESC>; any changes you have made but not saved to disk (using the "S" option from this screen) will be in effect only for the current ProComm session.

Select a setup section by typing its number and pressing <CR>. The screen will clear and display the current settings for that section. To change a setting, enter the number of the setting you wish to change and press <CR>. Then make your change as described below. If, after selecting a parameter, you decide not to change it, you can retain the current value by simply pressing <ESC>.

If the parameter you have selected requires a string, type in the characters for the new string and press <CR>. Note that pressing <CR> without typing any characters will clear the setting; if you wish to retain the current string, press <ESC>.

Some parameters offer you a selection of valid values. Press the space bar to cycle among the available values. When the value you desire is displayed, press <CR> to accept it.

If the setting you are changing requires a number, enter the new number and press <CR>. ProComm will check to make sure that the number falls within a valid range. If you have selected a number outside that range, ProComm will inform you of the error and allow you to select another number.

After making your changes in a particular section, press <ESC> to return to the main SetUp Menu. If you wish to save your changes to disk, select "S" from that menu. If you choose the Save option, the current settings will become your power-up defaults. If you do not save the setup, the

changes you have made will be in effect for only the current ProComm session.

Modem SetUp

If you select option 1 from the main SetUp Menu, a screen similar to the following will appear:

```

-----+ MODEM SETUP +-----
1) Modem init string .... ATE0 S7=60 S11=55 V1 X1 S0=0!
2) Dialing command ..... ATDT
3) Dialing cmd suffix ... !

4) Connect string ..... CONNECT
5) No Connect string 1 .. BUSY
6) No Connect string 2 .. VOICE
7) No Connect string 3 .. NO CARRIER
8) No Connect string 4 ..

9) Hangup string ..... ~~~+++~~~ATH0!

10) Redial timeout delay . 30
11) Redial pause delay ... 2

-----
OPTION ->                                     ESC Exit

```

You may change these options by typing the appropriate selection number.

1) Modem init string

The modem initialization string is sent to the modem every time you begin a ProComm session, and again whenever you exit Host Mode. It can be up to 46 characters in length, and may contain carriage returns or control characters using the translate conventions described in Section 2. Modem initialization strings vary among different types of modems. The default, configured for Hayes or compatible modems, is 'AT E0 S7=60 S11=55 S0=0 V1 X1!', where

AT is the command prefix

E0 sets modem echo of commands off

S7=60 sets the maximum wait time for a carrier at 60 seconds

S11=55 sets touch tone spacing (and is very fast)

S0=0 turns off the modem's auto answer feature

V1 activates verbal result codes

X1 activates the extended result codes

! causes ProComm to send a CR to the modem at the end of the modem

initialization string.

Be sure to include that last "!" if you need ProComm to send a CR at the end of the command.

The initialization string may be tailored to suit your particular modem and needs. Consult your modem user's guide for details regarding the operation and available commands for your modem.

2) Dialing command

The dialing command is used to instruct the modem to dial a number. It is sent to the modem by the Dial (Alt-D) and Redial (Alt-R) functions, followed by the number to dial and any long distance codes you might specify. The default is 'ATDT' where

AT is the command prefix

D is the dial command

T turns on touch tone dialing (P is used for pulse dialing)

The modem dialing command may contain imbedded pauses, CRs, ESCs and CTRL characters using the conventions on string translation described in Section 2.

3) Dialing cmd suffix

The dialing command suffix is sent to the modem at the end of a dialing command to indicate that the command is complete. The default command is "!", which is translated as a CR, and should be correct for most modems.

A complete dialing command includes the Dialing Command, the telephone number to be dialed (and any associated codes or numbers), and the Dialing Command Suffix. A local call using tone dialing, for example, might look like 'ATDT 123-4567!' where

ATDT is the dialing command

123-4567 is the number being dialed

and

! (translated as a CR) is the dialing command suffix

4) Connect string

The connect string is the message your modem sends to ProComm to indicate that a connection has been made. The default is "CONNECT". ProComm uses this value during automatic redial (Alt-R) to determine when a connection has been made. Note that this parameter must be set correctly (including upper- or lower-case) for auto redial to work. Translation is in effect for this string. For example, if your modem sends '<LF><LF><CR>' to indicate CONNECT (as the IBM PCjr modem does), set the Connect String to '^J^J^M'. (A Ctrl-J is a LF, and a Ctrl-M is a CR).

5) - 8) Modem No Connect strings

These strings are also used by the auto redial feature. They should be set to modem messages that indicate that a connection was not achieved. The defaults are "BUSY", "VOICE" and "NO CARRIER"; a fourth message may be added by selecting option number 8. Again, these strings must be exact matches to the messages your modem displays, including upper and lower case. Some modems do not support these call progress monitoring features, although most support at least "NO CARRIER".

9) Hang-up string

The hang-up string is the command sent to the modem to cause it to hang up. When Alt-H (Hangup) is pressed, ProComm first attempts to hang-up by dropping the Data Terminal Ready (DTR) line. If that attempt fails (determined by checking Carrier Detect [CD]), ProComm will send the hang-up string to the modem. The default, set up for Hayes and compatible modems, is "~~~+++~~~ATHO!", where

~~~ is a 1 1/2 second delay

+++ is the modem escape sequence to drop into command mode

~~~ is another 1 1/2 second delay

AT is the modem command prefix

HO is the hang-up command

! sends a <CR>

The "~" and "!" characters are translated by ProComm, not the modem (see Section 2 on string translation for information about how to change these characters). The "+++" surrounded by the 1 1/2 second pauses causes the modem to go into command state, where the hang-up command is then issued. For more information see the Alt-H command, in Section 4.

10) Redial Timeout Delay

This command determines the number of seconds that ProComm will wait during Redial (Alt-R) for a Connect or No Connect response from the modem. If the timeout delay is exceeded before a modem response is received, ProComm will cycle and attempt another redial. You should be sure that your modem's timeout value is set higher than this timeout value, or strange results may occur.

This value may also be modified on the fly during a redial attempt. See Section 5 on Redialing for more details.

11) Redial Pause Delay

The redial pause delay is the number of seconds that ProComm will pause between each dialing attempt during a redial. The pause is provided to

let the modem settle down between calls. The default is 2 seconds, and the minimum pause time is 1 second.

Terminal SetUp

When you select the Terminal SetUp option (number 2), you will be presented with a screen similar to the following:

```
-----+  TERMINAL SETUP  +-----
1) Terminal emulation ... ANSI-BBS  10) Enquiry ..... OFF
2) Duplex ..... HALF          11) Break Length (ms). 350
3) Flow Control ..... NONE
4) CR translation (in) .. CR
5) CR translation (out) . CR
6) BS translation ..... DEST
7) BS Key definition .... BS
8) Line wrap ..... OFF
9) Scroll ..... ON

-----
OPTION ==>                                ESC Exit
```

Select a parameter to change by entering its number followed by <CR>.

1) Terminal Emulation

Terminal emulation means using your computer to act like a terminal. Use this parameter to select the terminal you wish to emulate. All of ProComm's terminal emulations are described in detail in Appendix A.

After selecting option 1, press the space bar until the terminal you wish to use is displayed. Then press <CR> to make that terminal the current emulation. Take care to insure that the rest of the settable terminal parameters match what is expected for the terminal type you select. The DEC VT100 is the most popular terminal emulation for our users. For general BBS work, we suggest ANSI-BBS.

2) Duplex

You can control the default duplex setting here. Your choices are full duplex (no echo) and half duplex (local echo). The system administrator of the system you are calling can tell you what is appropriate for a particular system.

In general, if what you type is not displayed on the screen, but should be, try going to half duplex. If you see two of every character you type (e.g., "tttwwoo ooff eevveerryy cchhaarraacctteerr"), change to full duplex. Duplex may also be changed on-the-fly using the Alt-E command.

3) Flow control

Flow control (sometimes called "handshaking") is a method computers use

to control the way they talk back and forth. It's similar to a traffic light in that it determines in which direction traffic can flow at what time. The only flow control that ProComm currently supports is XON/XOFF, so your choices for this parameter are XON/XOFF or NONE. If you wish to use flow control set this option to XON/XOFF; otherwise select NONE. ProComm performs XON/XOFF at the interrupt level. To prevent deadlock because of extraneous XOFFs caused by line noise, ProComm allows you to reset the XOFF flag. If you see a message on the status line that indicates that an XOFF was received, but you think it is not valid, press <ESC> to clear the XOFF flag and allow you to continue work.

4) CR translation (in)

This feature is used to set the power-up default for incoming carriage return translation.

ProComm needs a CR/LF sequence to correctly handle lines sent to it by a remote. The CR (carriage return) moves the cursor to the beginning of the current line, and the LF (line feed) moves it to the next line. Some systems send only a CR and expect you to provide the line feed, while others send both the CR and LF.

When set to CR, ProComm leaves incoming carriage returns alone, and assumes that the remote system will also send a LF. When set to CR/LF, ProComm automatically adds a LF to any CR received. If the lines coming from the remote write on top of each other, you need to set this translation to CR/LF. If lines appear to be double spaced, go to CR.

Most TTY (non-full screen) applications (including most BBS's) send both the carriage return and line feed, so the ProComm default is CR. You can change this setting on-the-fly without affecting the power up default by using the Alt-F3 command.

5) CR translation (out)

Just as different systems send different line end sequences, they also may need to receive different line end sequences. Some systems need only a CR, while others must have a CR/LF combination.

Use this option to set the ProComm power up default for outgoing carriage return translation. If set to CR, outgoing carriage returns send only the CR. If set to CR/LF, any carriage return sent by ProComm to the remote unit will have a line feed appended to it.

The ProComm default is to send the CR only.

6) BS translation

A backspace (BS) may need to be interpreted differently depending upon the terminal emulation in use. In most cases it is "destructive" (DEST); that is, the cursor will both move to the left and delete the character in that position when the BS key is pressed or a BS is received from the remote. In other cases, the VT100 for instance, the BS behaves like a cursor-left command, merely moving the cursor without erasing any characters; it is thus "non-destructive" (NON-DEST). You may choose the

correct behavior for your application using this parameter. The ProComm default is DEST.

7) BS key definition

Normally, when the backspace key is pressed it sends a BS (ASCII 8) to the remote unit; by using this option, however, you can force ProComm to transmit a delete (DEL, ASCII 127) instead. The BS key definition option is especially useful when the terminal being emulated has a Del key where the IBM-PC has its BS key.

The ProComm default is to send a BS.

8) Line wrap

If an incoming line is greater than 80 characters long, it can be handled in two different ways. It may be truncated (cut off) so that characters past the 80th are lost, or it may be wrapped around to continue on the next line. This setting controls which method ProComm will use.

If line wrap is set ON, lines greater than 80 characters in length will wrap around and be displayed on the next line. With line wrap OFF, however, lines greater than 80 characters in length will be truncated.

Since most terminals truncate lines, the ProComm default is line wrap OFF.

9) Scroll

The scroll parameter controls what happens if ProComm receives a CR (or CR/LF) while the cursor is positioned at the bottom line. If the scroll option is set ON, ProComm moves all the lines on the screen up one line (losing the top line) and the new line is printed in the blank space at the bottom of the screen. If scroll is OFF, the cursor returns to the far left column, and the new line overprints the old. Normally this feature will be set ON (the default), although for some full screen applications you may need to inhibit screen scrolling.

10) Break Length (ms)

A break is a spacing condition on the line. It is often used to get a remote system's attention. ProComm uses Alt-F7 (or CTRL-BREAK on IBM machines) to signal a break.

The break length option allows you to set the length (in milliseconds) of the break signal. The default is 350 ms, and is sufficient for most systems.

11) Enquiry (Ctrl-E)

Some systems send an ENQ control character (Ctrl-E, ASCII 5) and expect an identifying sequence to be returned. If Enquiry is set ON, ProComm will respond to an ENQ by sending the keyboard macro assigned to Alt-O.

If Enquiry is set OFF, the ENQ will be treated as just another character.

A third option is available for users of the CompuServe Information Service (CIS). CompuServe uses an ENQ to signal the beginning of an automatic file transfer. If you wish to be able to perform automatic file transfers using the CompuServe 'B' protocol while logged on to CIS, set this option to CIS B. Be sure, however, that you are not set to CIS B while logged onto other systems, or strange results may occur.

The ProComm default for Enquiry is OFF.

Kermit SetUp

The Kermit Setup section provides control over a number of Kermit file transfer parameters. After selecting option 3 from the main SetUp Menu, you will be presented with a screen similar to the following:

```
-----+ KERMIT SETUP +-----
1) Control quote char ... 35  (ASCII)
2) Maximum packet size .. 90
3) Pad character ..... 0    (ASCII)
4) Number of pad chars .. 0
5) 8th bit quote char ... 38  (ASCII)
6) Handshake char ..... 0    (ASCII)
7) End of line char ..... 13  (ASCII)
8) File type ..... BINARY
9) Block check type ..... 1 BYTE CHECKSUM

-----
OPTION ->                                     ESC Exit
```

We do not have the space here to give a Kermit tutorial. If you do not know what these elements are, you probably should not change them. Consult the system administrator for your system if you have any questions regarding their Kermit installation.

One parameter worth mentioning is the Handshake character (option 6). In most implementations you should use the default of 0, which implies no handshaking. In the case of line at a time (not full screen) IBM mainframe access in half duplex, a handshake value of ASCII 17 is appropriate.

A further discussion of ProComm's Kermit implementation is available in the file transfer section, Section 6.

General SetUp

Selecting the General Setup option from the main SetUp Menu will cause the following screen to appear:

-----+ GENERAL SETUP +-----

```

1) Editor name .....
2) Default d/l path ....
3) Default log file ..... PROCOMM.LOG
4) Screen dump file ..... PROCOMM.IMG
5) Screen write method .. DIRECT
6) Translate table ..... OFF
7) Sound effects ..... ON
8) Alarm sound ..... ON
9) Alarm time (secs) .... 5
10) Exploding widows ..... YES
11) XMODEM mode ..... NORMAL
12) Xlat pause character . ~
13) Xlat CR character .... !
14) Xlat CTRL character .. ^
15) Xlat ESC character ... |
16) Aborted downloads .... KEEP
17) Transmit pacing (ms) . 30

```

OPTION =>

ESC Exit

1) Editor name

Use this parameter to name the program to be accessed by the Alt-A (Editor) command. Pressing Alt-A will execute this program from within ProComm, without the necessity of popping out through the DOS Gateway or exiting ProComm. This is very handy for editing or listing a file while online, as well as for developing ProComm command files and many other tasks.

The Editor name parameter can be specified in a number of ways. It may be a complete filename with path (e.g. C:\WP\EDIT\EDITOR.EXE), just the program name without extension (e.g. EDITOR) or anything in between. If no path is specified, ProComm will search the directories specified in the environment variable PATH for the indicated program.

ProComm can run almost any .EXE or .COM program using this feature as long as there is enough memory available. Keep in mind that ProComm itself requires approximately 130K of RAM, and the operating system another 17-30K (depending on the version). Other resident programs (such as SideKick) can take up even more. If you are running only 192K (the minimum possible to operate ProComm), you may not have enough memory to load the desired program unless it is quite small.

The Alt-A command cannot be used to execute a batch (.BAT) file.

2) Default d/l path

This option allows you to specify where you want to put files that you download. If no path is specified, downloads will be directed to the currently logged drive and directory. If a path is specified, the file will be placed in the named directory. For example, if this option is set to

```
C:\COMM\PROCOMM\DL\
```

then all files that you download will be placed in the \COMM\PROCOMM\DL directory on drive C:. Keep in mind that the pathname must end with a backslash ("\") in order for ProComm to interpret it correctly. Consult your DOS user's guide for details about directory naming conventions.

You can override the download directory option when using the file transfer protocols which require you to name the incoming file (XMODEM, YMODEM, and ASCII). To do so, include a path when specifying the filename to download. For example, when downloading a file using the XMODEM protocol, ProComm will prompt you for the name of the file to be downloaded. If you specify

C:\BASIC\FILENAME.EXT

as the filename, the file will go directly to the \BASIC directory on the C: drive, and not to the default download directory. The other file transfer protocols include the filename as part of the data being sent, so you will not be prompted for the filename to use and thus cannot override the default download directory option.

3) Default log file

When you activate file logging (Alt-F1), ProComm will prompt you for the name of the log file to use. Pressing <CR> without naming a file directs the log to the default file. Use this parameter to set the default name for the log file to use when file logging is in effect. If the file exists, ProComm will not overwrite existing data; rather, new data will be appended to the end of the file. If the file does not exist it will be created in the current directory and the data will be saved to it.

4) Screen dump file

This option names the file to which ProComm will append screen dumps (Alt-G). If the screen dump file does not exist when Alt-G is pressed, it will be created in the current directory. Again, ProComm will not overwrite an existing file, but will append the screen dump data to the end of an existing file.

5) Screen write method

ProComm can use either of two different methods to perform its screen writing. In the first method, characters are written directly to the screen buffer memory area. In the second, characters are written using BIOS (operating system) function calls. Normally you would use direct screen writes because they are much faster. In some cases, however, such as under multi-tasking operating systems or when you are using a not-so-compatible computer, you might wish to use the BIOS. The ProComm default is to use direct screen writes. You may also select BIOS mode using the "/M" command line option.

6) Translate table

This setting controls whether or not ProComm uses the translation table to translate, or strip, incoming characters. Selecting YES causes the translate table to be activated immediately, as well as on program startup. ProComm defaults to NO. The translate table may be defined and toggled ON/OFF using the Alt-W command from the Terminal Mode or from the Help Screen. The translate table is discussed in more detail later in

this manual.

7) Sound effects

This option controls the use of ProComm-produced sound effects. These sound effects include the open and close window sounds, as well as some other audible feedback. It does not control the alarm function, sounds generated by the modem or beeps (Ctrl-G, ASCII 7) sent from the host unit. Sound effects are ON by default, and may also be controlled using the "/S" command line option (see Section 2).

8) Alarm sound

This setting controls whether or not the alarm is audible. The alarm is used to indicate the end of file transfers, connects during re-dials and other events. It may also be activated by the ALARM script command. If the alarm sound is ON, these events will trigger a ringing sound which will continue for the number of seconds specified by the alarm time setting. If the alarm sound is OFF, the appropriate message will flash for the indicated time but will be silent. By default, alarm sound is ON.

9) Alarm time (secs)

Use the alarm time setting to determine the time (in seconds) that you wish the alarm sound to ring. If, for example, you want the file transfer alarm to ring for two minutes, then set the alarm time to 120.

10) Exploding windows

ProComm makes extensive use of windows in its operation. This option controls the manner in which those windows appear. If you select YES, ProComm will use "exploding" windows, i.e. windows that start small and rapidly grow to full size. If, on the other hand, you select NO, then windows will appear without expanding frames. By default ProComm will use exploding windows. This is purely a cosmetic effect, and has no bearing on the functionality of the program.

11) XMODEM mode

Some remote systems, such as CompuServe, cannot handle the normal XMODEM error timeout periods. They need a less critical timing situation. For these systems, set the XMODEM mode to RELAXED to avoid file transfer aborts resulting from timing errors. In most cases, however, this option should be set to NORMAL.

12) Xlat pause character

The translate pause character parameter is used to set the character which is to be interpreted as a pause during string translation. ProComm uses the tilde (~) as the default. See Section 2 for more details on string translation.

13) Xlat CR character

This parameter sets the character which is to be interpreted as a carriage return during string translation (the translate CR character). By default, an exclamation point (!) is used. See Section 2 for details about translation characters.

14) Xlat CTRL character

The translate CTRL character parameter is used to set the character which is interpreted as signaling a control character during string translation. The default is the caret (^). Section 2 has more details.

15) Xlat ESC character

Use this parameter to set the character which is to be interpreted as an ESC during string translation. By default, ProComm uses a vertical bar (|). See Section 2 on string translation for more details.

16) Aborted downloads

This setting will determine the disposition of files that are aborted during downloads. If it is set to KEEP, these partial files are kept on the disk and are your responsibility. If it is set to DISCARD, they are erased from your disk when the download aborts. By default, aborted downloads are kept.

17) Transmit pacing (ms)

This parameter controls output pacing of strings. Strings that are paced include all the setup strings, macro keys, and terminal control sequences such as function keys and cursor control. This option is provided for those systems too slow to handle ProComm's speed.

Host Mode Setup

Selecting the Host Mode Setup option from the main Setup Menu will cause the Host Mode Setup screen to appear:

```
-----+ HOST MODE SETUP +-----  
  
1) Host ID string ..... Welcome to ProComm Host!  
2) Auto answer string ... ~~~+++~~~ATSO=1!  
3) Host mode password ... PASSWORD  
4) DOS shell password ... SHELL  
5) Auto baud detect ..... METHOD 2  
6) Connection type ..... MODEM  
  
-----  
OPTION =>
```

ESC Exit

1) Host ID string

The host ID string is a message that is sent to a remote caller when he connects to ProComm in Host Mode. It can be set to anything you desire. Notice that the default message includes a CR translation character (here the "!") at the end of the string.

2) Auto answer string

The auto answer string can be thought of as a modem initialization string for use with the host mode. Use it to set the modem into auto answer mode. The default string is "~~~+++~~~ATSO=1!", where

~~~ is a 1 1/2 second pause

+++ sets the modem in command state

~~~ is another 1 1/2 second pause

AT is the command prefix

SO=1 sets the modem to answer after one ring

! causes ProComm to send a CR.

The "~~~" part is a ProComm command rather than a modem command and will probably not have to be reset. The other commands depend on the needs of your particular modem. Consult your modem user's guide for further information.

3) Host mode password

ProComm Host Mode provides some security by means of access passwords. The host mode password must be correctly entered by all remote callers before they are granted access to your system. The caller must match the password completely, including upper- and lower-case letters. If you set the password to null, then callers can get through the password prompt by pressing <CR> without typing anything else.

4) DOS shell password

Callers must know this secondary password to be allowed access to the remote DOS shell. Be very careful about this password; you do not want just anyone to have system-level access on your machine. You should definitely not leave the DOS shell password blank, nor should you leave it at the default.

5) Auto baud detect

Auto baud detect allows ProComm to match the baud rate at which a user calls.

ProComm provides three choices of auto baud detect in host mode. The first choice is NONE, that is ProComm will not attempt to match baud rates with incoming calls. Users must call at the same rate that ProComm is set to in order to be connected. The second choice, called MODEM MSG, uses modem messages to determine baud rate. Your modem must support the messages CONNECT (for 300 baud), CONNECT 1200 and CONNECT 2400 in order to use MODEM MSG. In addition, your modem must be configured to return these message. That configuration is usually achieved via the Xn modem command, which may be placed in either the modem initialization string, or the auto answer string. The third choice, known as KEY HIT, requires that callers enter several <CR>s (or SPACES if at 2400 baud) in order for ProComm to match baud rates. No modem messages are required. ProComm defaults to KEY HIT.

6) Connection type

The connection type determines who ProComm Host Mode is talking to. When set to MODEM, ProComm assumes a modem connection and performs as described in the Host Mode section later in this manual. When set to DIRECT, ProComm bypasses the carrier detect process and immediately initiates a connection; this feature is useful for direct connecting to other computers.

ASCII Transfer Setup

The ASCII Transfer Setup Menu is used to determine the characteristics of file transfers in the ASCII mode:

```
-----+ ASCII TRANSFER SETUP +-----
      ASCII UPLOAD
1) Echo locally ..... NO
2) Expand blank lines ... YES

3) Pace character ..... 0   (ASCII)
4) Character pacing ..... 15 (1/1000 sec)
5) Line pacing ..... 10  (1/10 sec)
6) CR translation ..... NONE
5) LF translation ..... STRIP

      ASCII DOWNLOAD

8) CR translation ..... NONE
9) LF translation ..... NONE

-----
OPTION =>                                     ESC Exit
```

ASCII UpLoad

1) Echo locally

Use this setting to control whether or not ProComm echoes locally what it is transferring during ASCII uploads. In most cases set this to NO and let the remote do any desired echoing. If the remote and ProComm are both displaying what is being transferred, you'll have a real mess on your screen.

2) Expand blank lines

Many systems interpret a blank line to mean "end of text". This is especially true when entering online messages. Use this option to tell ProComm to expand blank lines. This would allow you to include blank lines (for spacing) in messages that you are uploading without the remote thinking it is the end of the message. When set ON, ProComm will add a space to lines being uploaded that contain only a CR or CR/LF. When set OFF, lines are uploaded as they exist. By default ProComm will expand blank lines.

3) Pace character

The Pace character provides one means of pacing uploaded text. If the pace character is set to a value other than 0, ProComm will send a line, then wait to receive the specified character before sending the next line. Enter the decimal value for the ASCII character desired; for example, set it to 13 to indicate a carriage return. ProComm will send a line and then wait to receive a CR from the remote before continuing to send the next line. The default is 0, which means that no pace character is used.

4) Character pacing

Another pacing option that ProComm provides is character pacing. Specifying this option causes ProComm to pause after each character has been sent during an ASCII upload. This can help avoid over-running the remote computer's input buffer. If character pacing is set to a value other than 0, ProComm will send a character, then wait the specified number of milliseconds (1/1000 second) before sending another character. By default, character pacing is set at 15 ms. Character pacing may be used in conjunction with any of the other pacing options.

5) Line pacing

ProComm can also perform line pacing during ASCII uploads if you so desire. Line pacing is similar to character pacing except that the pause occurs after each line, rather than after each character. After a line is sent, the program will wait for the time specified (in 1/10 seconds) before sending the next line. Pacing may be set to zero if the remote can handle that speed of transmission. Line pacing may be used in conjunction with character pacing; it is generally not needed if you are using a pace character. The ProComm default for line pacing is 10-tenths of a second (i.e., one full second).

6) CR translation (uploads)

As discussed above, different systems require different line end sequences. This option controls outgoing carriage return translation during an ASCII upload. You have 3 options. If CR translation is set to NONE, no translation is performed and carriage returns are passed directly to the remote. If set to STRIP, all carriage returns encountered in the file being uploaded are stripped, and not sent. Finally, if set to ADD LF, a line feed will be added to all outgoing carriage returns. ProComm defaults to NONE.

7) LF translation (uploads)

LF translation is similar to CR translation, but affects line feeds. As with CR translation, you have 3 options. If you select NONE, no translation is performed. If you set LF translation to STRIP, all line feeds encountered in the file being uploaded are stripped. If you set it to ADD CR, a carriage return will precede all outgoing line feeds. The default for outgoing LF translation is STRIP.

You should pay close attention to what is being done with these parameters. On a PC, most text files are delimited by a CR/LF sequence. Most mainframe and other systems, however, want to receive ASCII files with only a CR as the delimiter. Thus the ProComm defaults take a file containing CR/LF sequences and transmit it as a CR-only delimited file. The various combinations available in ProComm will allow you to transfer ASCII text files to virtually any system.

ASCII Download

8) CR translation

The download CR translation is exactly like that described above only it applies to text going in the other direction; it translates CRs coming in to ProComm from the remote. There are three options for controlling incoming carriage return translation during ASCII downloads. If CR translation is set to NONE, no translation is performed. STRIP causes all carriage returns encountered in the file being received to be stripped, while ADD LF causes a line feed to be added to all incoming carriage returns. The ProComm default is NONE.

9) LF translation

This option controls incoming line feed translation during ASCII downloads. It also has 3 options. If it is set to NONE, no translation is performed. If it is set to STRIP, all line feeds encountered in the file being downloaded are stripped, and if it is set to ADD CR, a carriage return will be added in front of all received line feeds. NONE is the default translation.

Once again the variety of ASCII translation options will allow you to receive ASCII text files from any system in a format suitable for your PC.

4. MAJOR FUNCTIONS

After the opening display and the program initialization, you are left in Terminal Mode. The bottom line of the screen is a status line, but the rest of the screen is open for use. Most of your communicating takes place in Terminal Mode, and most ProComm commands are executed from there as well. Commands are usually executed in a window which leaves your original screen intact.

Commands are activated by pressing certain keystrokes while in Terminal Mode. You may, however, get a listing of available commands by pressing Alt-F10, the Help key. You can then execute commands directly from the Help Screen without returning to the Terminal Mode. Pressing any non-command key will return you to Terminal Mode.

The Help Screen lists all of the major ProComm features, divided into three blocks: Major Functions, Utility Functions, and File Functions.

Major Functions

Dialing Directory

Pressing Alt-D activates the dialing directory. The dialing directory is an online phone list which contains up to 100 entries and various information relating to those entries such as baud rate, parity, echo, etc. The functions available from the dialing directory are described in detail in Section 5.

Automatic Redial

The automatic redial facility (accessed with Alt-R) provides for automatic redialing of a single number or a list of numbers. It is especially useful for connecting to hard-to-reach numbers. Its features and how to use them are examined in Section 5.

Keyboard Macros

Keyboard macros allow you to assign character strings to the keys Alt-0 through Alt-9, which may then be used to transmit the assigned string to the remote with a single keystroke. Macro strings may be up to 50 characters long, and may contain imbedded control codes and carriage returns using the translate conventions described in Section 2. To send the string you've assigned, simply press the appropriate key.

Access the keyboard macro facility by pressing Alt-M from Terminal Mode or the Help Screen. A window will appear listing the current key assignments. To revise an assignment press "R", then the key to assign

(Alt-0 through Alt-9). Now type the string you wish assigned to that key. Press <CR> when you are done. Respond "Y" to the 'OK' prompt to accept the assignment; otherwise, it is discarded. After creating a group of macros be sure to save them to disk, using the "S" option, or they will be discarded when you leave the current ProComm session. Press <ESC> to return to Terminal Mode.

You may create and use any number of keyboard macro files for use with different systems. By default, ProComm will load the file called PROCOMM.KEY when it is brought up. To create other .KEY files, first select "C" (clear) to clear out the current macro definitions. Then use the "R" (revise) option as described above to create your new set of definitions. Now choose "S" to save the new definitions. When prompted for a file name, give the name of the file you wish the macro definitions to be saved in. The file may use any valid filename; we suggest using a file extension of .KEY to help distinguish the nature its contents. Pressing CR without naming a file will cause ProComm to store the macros in the default file PROCOMM.KEY.

To load a new macro file select option "L" (load). Then give the name of the macro file to load. Again, if you press <CR> without naming a specific file, ProComm will use the default (PROCOMM.KEY). The named file will be loaded and the new definitions displayed. Macro key files may also be loaded via the MLOAD script command.

Multiple macro key files give you considerable flexibility in using different systems. One thing you can do is create different .KEY files for various systems that you call, then load the macros via a MLOAD command in a script file linked to that system in the dialing directory. If you use a consistent scheme (i.e. user ID in Alt-1, password in Alt-2) you can simplify many of your online tasks.

Line Settings

ProComm supports a wide variety of communications configurations. You may select baud rate, the number of data and stop bits, parity and duplex. The program defaults to 300 baud operation, with 8 data bits, 1 stop bit, no parity and echo off (full duplex).

Use the Alt-P command to review or change the active line control settings. After opening the Line Settings window, ProComm displays the current settings. Enter a number from 1 to 12 to change to a predefined setting. Use the numbers 13 through 19 to define a customized configuration. Enter 20 through 23 to make COM1 through COM4 your active port.

The save option (24) is used to make the new configuration your default. If you change the line settings without saving them to disk, they will be used only for the current session.

Press <ESC> to return to Terminal Mode, and the new settings will be in effect.

Various systems have different line control needs. By far the most common settings are N/8/1 (no parity, 8 data bits and 1 stop bit) and E/7/1 (even parity, 7 data bits and 1 stop bit). Most bulletin board systems (BBS) require N/8/1. Many mainframe computers use E/7/1. When

calling online services such as CompuServe and The Source via public networks like Telenet and Tymnet, be sure to use E/7/1. A good rule of thumb is if you are at N/8/1 and your screen displays a lot of garbage (graphics characters and the like), switch to E/7/1.

An alternative to using E/7/1 on some systems is to use N/8/1 and strip the high bit off all incoming characters using the translate table. To do so, set the upper 128 (128-255) characters in the translate table to a value exactly 128 less than their ASCII decimal values. Thus 128 would be set to 0, and 255 would be set to 127.

Translate Table

ProComm's translate table provides a means for you to strip or replace unwanted characters you receive from the remote. Press Alt-W to display the current translation settings. The table will indicate whether translation is currently taking place, as well as showing you the current translation values for the ASCII characters 0-127. Press F2 to display translation values for characters 128-255. Pressing the keys F3 and F4 will toggle the translation effect on or off. You may set the default condition from the General Setup screen in the Setup menu (Alt-S).

To change a translation value, enter the decimal ASCII code to reset. Now enter the translated value (again in decimal). To strip an unwanted character, translate it to 0. For example, suppose the system you are calling sends a lot of beeps (ASCII 7) and you want to filter them out. Bring up the translate table (Alt-W). Enter a 7 at the 'NUMBER TO CHANGE' prompt, and then a 0 at the 'NEW VALUE' prompt. The change will be highlighted in the display. Now save the table by pressing F1, and activate it by pressing F3. All beeps (ASCII 7) coming from the remote will now be stripped out. Use the same technique to translate a given value to something else. Press <ESC> to return to Terminal Mode.

One important note: when you turn the translate table ON, it reads the .XLT file from disk and overwrites the existing table. Thus you cannot make some changes, then turn on the table, since the changes you made will be overwritten. You must either make the changes, save the changes and then turn on the table, or turn on the table, and then make your changes. To make the changes effective for only the current session, do not save the changes to disk.

Editor

To call an editor, word processor or other program from within ProComm, press Alt-A. ProComm will then attempt to load the program you listed as the editor name in the General Setup portion of the Setup screen. You can specify any executable program (except batch files) to be called by the Alt-A command. This command is quite useful for viewing a file, doing some editing, or whatever.

In order for this function to work, however, a couple of things must be correctly set. First, COMMAND.COM must reside on the drive you booted from. Second, ProComm must be able to find the program you requested. (See the discussion on the editor name in the General Setup section). Third, your computer must have enough available memory to execute the desired program. There is no default for this feature, so be sure to set

it up before attempting to use it.

Exit

To exit ProComm, press Alt-X. You will be asked to verify your decision to assure that you are not exiting by mistake. Answer "Y" to exit, "N" to continue working. Be sure that you have saved any setup changes you have made and wish to keep before exiting or they will be lost. Similarly, be sure you have completed your online tasks and signed off, since ProComm will hangup when you exit.

Host Mode

ProComm includes a limited Host Mode which allows remote access to your computer. You can activate the Host Mode by pressing Alt-Q. Host mode features password protection, file transfers, operator page and DOS shell access. It can also display a canned message or graphics screen, and maintain a history of logons. Host Mode is described in detail in Section 8.

Chat Mode

Chat Mode provides split screen operation for online conversations. Incoming text (and echoed outgoing text if in full duplex) is displayed in the top 18 lines of the screen. Outgoing text is displayed in the bottom 4 lines.

Activate Chat Mode by pressing Alt-O. Limited line editing is provided for outgoing text; use the backspace key to edit a line before it gets sent. Text is sent to the remote only after a carriage return or when the buffer gets full (about 3 full lines of text). Ctrl-Q and Ctrl-S keystrokes (XON and XOFF), however, are sent immediately.

Printer and disk logging will continue if they are active when Chat Mode begins. The Redisplay facility (Alt-F6) is also available in Chat Mode. Press <ESC> to exit Chat Mode and return to the normal Terminal Mode.

DOS Gateway

ProComm provides a gateway to DOS which allows you to execute DOS commands or other programs without ending the ProComm session. To activate the DOS gateway press Alt-F4. ProComm uses the COMSPEC environment variable to find the command processor, so make sure that COMMAND.COM is present on the boot drive, or the gateway will not work correctly. Type "EXIT" on the DOS command line when you wish to return to ProComm.

CAUTION: executing other communications programs through the Gateway may cause erratic results when you return to ProComm. If this occurs, using the Alt-P command to reset ProComm's line settings may re-establish the connection.

Command Files

Command files are text files containing ProComm commands. You can use command files to perform automatic logons, unattended file transfers, and many other tasks. You can create command files using virtually any word processor provided that the program can save files in a "non-document"--or straight ASCII--format. Command files may be executed on program startup, from the command file menu (Alt-F5), or by linking them with dialing directory entries. See Section 7 for a complete discussion of command files and the ProComm command set.

Redisplay

To redisplay lines that have scrolled off your screen, press Alt-F6. ProComm will display the last 10000 characters that have come in, beginning with the most recent screen.

You can move through the redisplay buffer in any of several ways: use the PgUp and PgDn keys to scroll one page in either direction; use the up and down arrow keys to move one line in either direction. Pressing the Home key will cause the first page of the buffer to display, while pressing the End key will display the last page. To search for specific text with the redisplay buffer, press "F" or "/". A window will open, and you will be prompted for the string to search for. If the string is found, ProComm will scroll to the page it is on and highlight it. To search for the same string again, press <CR> when ProComm prompts you for the text to look for. Searches are not case sensitive.

Utility Functions

Program Information

To display the ProComm program information screen, press Alt-I. Type any key to return to Terminal Mode.

SetUp Screen

Use the SetUp option (Alt-S) to access the 6 setup areas: modem setup, terminal setup, Kermit setup, general setup, host mode setup, and ASCII file transfer setup. These options are described in detail in Section 3.

Kermit Server Command

ProComm provides several Kermit server commands for use with remote systems running in Kermit server mode. Access the command menu by pressing Alt-K, then select the desired command from the menu. See section 6 for more details regarding these commands.

Change Directory

You can use the Alt-B command to change the default directory and/or the active drive. To change directories, press Alt-B. A window will appear naming the current drive and directory. Simply enter the drive

(including a colon), directory or both and press <CR>. You have now changed the default drive and or directory. Press <ESC> to leave the default unchanged.

Clear Screen

Press Alt-C to clear your screen and home the cursor. This is a local effect only. Clearing the screen will also reset ProComm to its default colors, useful when connecting to systems which change colors and don't reset them.

Toggle Duplex

Pressing Alt-E will toggle ProComm between full and half duplex. You can set the default duplex in ProComm's SetUp (Alt-S) screen. If characters you type appear twice (e.g., "AApppeeaarr TTwwiiccee") you should set duplex to full. Similarly, if you type something that is not displayed, but should be, try toggling duplex to half. A message is briefly displayed on the status line, and the duplex block (the third block on the status line) indicates whether you are currently operating under half or full duplex ("HDX" or "FDX").

Hang Up Phone

Press Alt-H to hang-up your telephone connection. ProComm will first attempt to hangup by dropping DTR (Data Terminal Ready). If this attempt fails, as indicated by the presence of CD, then the modem hang-up string will be sent to the modem. If ProComm is not causing your modem to hang up correctly, be sure that your modem does not have DTR or CD forced high, and check the hang-up string in the modem SetUp screen (via Alt-S) against the string suggested by your modem user's guide.

Elapsed Time

Pressing Alt-T will display the current time and date, as well as the elapsed time since the last call was made. Elapsed time is reset every time you make a call using the dialing directory or make a connection using the automatic redial facility.

Print On/Off

Press Alt-L to toggle printer logging on or off. If printer logging is toggled on, any information coming from the remote system will be sent directly to your printer as well as to your screen. The status of the printer log is indicated by the message PRT ON or PRT OFF on the status line.

Set Colors

Enter Alt-Z from Terminal Mode to set your local screen colors. A window will open with a list of window selections on the left and instructions on the right. Press the up or down arrow keys to select a window to

change. Next, press the left or right arrow keys to choose which part of the window to change (background, foreground, or hi-light). Press the space bar to cycle among the color choices. After you have set all the colors you want to change, press <ESC> to exit. You will then be asked if you want to save the color changes to disk. If you save the changes to disk, they will remain until you change them again; otherwise they will remain only for the current session. You can use this last option to set the colors temporarily, try them out during a session, then return to Alt-Z and save them to disk.

Some windows do not appear in the window list but are settable anyway. The Alt-T (elapsed time), Alt-B (change directory), and Alt-X (exit) windows use the same colors as the keyboard macros window. The redial display (Alt-R) uses the phone directory (Alt-D) colors. The SetUp screens (Alt-S) use the help window colors. Communication screen colors are used in the redisplay section and the command file selection window (Alt-F5) uses the file transfer window colors.

Auto Answer

You may control the auto answer feature of most modems using a software command. For instance, sending the command "ATSO=1<CR>" to a Hayes modem tells it to answer the phone on the first ring. Similarly, the command "ATSO=0<CR>" tells that modem to turn auto answer off.

ProComm allows you to issue an auto answer command at the touch of a keystroke. Pressing Alt-Y causes the string you have specified as the auto answer string in the Host Mode section of the SetUp screen to be sent to the modem. Note that this command is used to set the modem into auto answer mode; it is not used to command the modem to answer immediately. You could use a command such as "ATA!" in a keyboard macro to instruct the modem to answer on command. See Sections 3 and 8 for more information regarding the auto answer string.

Toggle CR - CR/LF

Some remote systems delimit lines of text with a single carriage return, while others use a carriage return/line feed combination. ProComm likes to receive lines that end with both a CR and a LF. If necessary, ProComm can add a line feed to each incoming carriage return.

Use Alt-F3 to toggle this feature between CR and CR/LF. The default is CR only, that is, do not add additional LFs. If text lines overwrite each other, the remote is not sending LFs, so switch to CR/LF. If text appears double spaced, try the CR setting. A message is briefly displayed when you toggle this option, and the current setting is reflected in the next-to-last block of the status line. The default condition for this option may be set in the Terminal Setup section of the SetUp facility.

Break

A break is a spacing condition on the line, often used to signal attention to the remote. Pressing CTRL-BREAK will cause the break condition to occur. On some non-IBM machines pressing CTRL-BREAK can

cause a lockup; therefore ProComm provides another keystroke, Alt-F7, which also causes the break condition to occur. The default time period for a break is 350 milliseconds, but that may be changed via the Terminal SetUp screen. Use the keyboard macro feature if you require that a specific character be sent in order for the remote to realize a break.

File Functions

Send Files (Upload)

ProComm provides a number of common protocols for use transferring files; pressing PgUp will activate the screen listing those protocols and allow you to choose among them. See Section 6 for a detailed description of file transfers.

Receive Files (Download)

Pressing PgDn activates the download menu screen and allows you to select from the available protocols. See Section 6 for more details on file transfers.

Directory

Press Alt-F to get a file listing of the currently logged drive or directory. Enter the file specification ("filespec") at the prompt just as you would in the DOS DIR command. Paths and drives are supported. COMSPEC must be set correctly for this feature to work. That is, you must have COMMAND.COM on the boot drive, or the Alt-F command will not work correctly.

View a File

To examine a file that exists on your system type Alt-V. You will be prompted for the name of the file. Enter the complete filename, including drive and path designator if desired. Use the PgUp and PgDn keys to page through the file. Press Home to go to the beginning of the file. Pressing <ESC> will end the viewing procedure and return to the Terminal Mode. This viewing facility is rather primitive; it is intended only as a convenience. For more complete browsing power, use the Alt-A editor command, or drop through the DOS gateway and use your favorite list program.

Screen Dump

Pressing Alt-G activates a screen dump of the current screen contents. Screen contents are appended to the current .IMG file, which may be specified in the SetUp screen. The default screen dump file is PROCOMM.IMG.

Log Toggle / Log Hold

ProComm provides two alternatives to file transfer for capturing information: file and printer logging. After initiating the logging procedure, any information coming from the remote system is routed to the log as well as to the screen.

Press Alt-L to begin capturing information to your printer. A message on the status line will indicate that the printer log is open. Pressing Alt-L again will toggle printer logging off.

ProComm writes the logged data to the DOS device PRN. The default value for PRN is LPT1:. You may use the DOS 'MODE' command to redirect PRN. For example, if your printer is on COM1: use the DOS command

```
MODE LPT1:=COM1:
```

to send printer output there.

Data can also be captured to disk, using the Alt-F1 command. Specify the name of the file to be used, or press <CR> to use the default file. If the file already exists, new information will be appended at the end of the file. Press Alt-F2 (Log Hold) to suspend information capture without closing the log file. A message indicating logging status will be displayed on the bottom line of the screen. Press Alt-F1 again to toggle the log file closed.

With the exception of formfeeds, terminal control sequences are not included in the log files.

5. D I A L I N G D I R E C T O R Y

The ProComm dialing directory automates most of your dialing tasks. The directory holds information for 100 of your most often-called systems. The information includes the name and phone number, as well the communications parameters specific to that entry. Also included in the directory is a user definable modem command, with an optional suffix, and up to four long distance codes that you can use with alternate long-distance services. You can even link command files to dialing directory entries to provide custom setup configurations or to perform logon sequences automatically.

Press Alt-D to access the dialing directory. A window is opened that displays pages of 10 entries at a time:

```

+-----+ D I A L I N G   D I R E C T O R Y +-----+
|
|      Name                      Number      Baud P D S   E   CMD File
|
|  1- ProComm Support BBS      1 314 449-9401 2400-N-8-1   N
|  2- .....                    . . . . . 1200-N-8-1   N
|  3- .....                    . . . . . 1200-N-8-1   N
|  4- .....                    . . . . . 1200-N-8-1   N
|  5- .....                    . . . . . 1200-N-8-1   N
|  6- .....                    . . . . . 1200-N-8-1   N
|  7- .....                    . . . . . 1200-N-8-1   N
|  8- .....                    . . . . . 1200-N-8-1   N
|  9- .....                    . . . . . 1200-N-8-1   N
| 10- .....                    . . . . . 1200-N-8-1   N
|
|  ==>  R Revise                M Manual Dialing      Entry to Dial
|        P LD Codes              D Delete Entries      F Find
|        PgUp/PgDn Page          L Print Entries      / Scroll
|        Home Top Page           End Bottom Page       ESC Exit
|
|        Modem Dial Cmd:  ATDT                LD Codes Active:
|        Dial Cmd Suffix:  !                  Com Port Active: COM1
|
+-----+

```

The current modem command ("Modem Dial Cmd"), modem command suffix ("Dial Cmd Suffix"), active long distance codes ("LD Codes Active") and the active port ("Com Port Active") are displayed at the bottom of the screen. The modem command defaults to "ATDT", with "!" (translated as a CR) as the suffix.

- Use the PgUp and PgDn keys to display the previous or next page of entries. The up and down arrow keys will scroll the display one entry in either direction. The Home key will position the display at the first page in the directory, while the End key will position it at the last page. Press <ESC> to exit the dialing directory.

Searching for an Entry

The dialing directory also has a search capability. To look for a particular entry press "F" or "/". A window will open asking for the text to search for. Enter the string you wish to be found and press <CR>. ProComm will search the name and number fields for the string you provided. If the string is found, the dialing directory will scroll to the page containing that entry, and the entry will be highlighted.

The dialing directory search facility is not case sensitive. For example, a search for the string "abcd" will match "abcd", "ABCD", "AbCd" or any other mix of upper- and lower-case characters. You can even search for telephone numbers or even parts of numbers. (That is, you can search for "Phil Sidney", just "Sidney", or "555-1586", or even "1586".) If the search routine reaches the end of the dialing directory without finding a match, it will wrap around to the beginning of the directory and continue its search until it has checked each entry once. If the string is still not found, a message to that effect is displayed. To repeat a search for the same text, press "F" or "/" a second time and then immediately press <CR>.

Revising the Dialing Directory

When you first use the dialing directory most of the entries will be empty. The name and phone number fields will be filled with periods; the default baud rate is 1200, with no parity, 8 data bits, 1 stop bit and echo off (full duplex). You may add a new entry or revise an old one by selecting the "R" option. You may also revise the long distance codes (by entering the appropriate LD code identifier: -,+,@,#). An entry need not be displayed on the screen in order to revise it.

Adding or Revising an Entry

To add an entry to the directory, or to change an existing entry, enter "R" at the "==" prompt. A window will open and you will be asked for the entry to revise. Type the entry number (1-100) you wish to revise. The current values for that entry are displayed at the top of the window. You are prompted first for the name. Press <CR> to retain the current value, or enter up to 24 characters for a new or revised name. You can use the backspace and left arrow keys to edit your entry. Press <CR> when you are done. You will now be prompted for the phone number. Enter only the numbers you need; the number will be right justified on the page. For example, if you are entering a local number, you need not enter spaces for the area code. Press <CR> when you are finished entering the number.

The current baud rate will now be displayed. Press any key (except <ESC> or <CR>) to cycle through the available baud rates. Press <CR> to select the one you want. Use the same technique to select the parity, data bits, stop bits and echo. You will now be prompted for a command file to link to the entry. Enter the filename without an extension or path. The same command file may be linked to any number of dialing directory entries. To clear an existing command file, type a space and then <CR>. If you don't want to link a command file with this entry, then press <CR> without specifying a filename.

After specifying the command file, you will be asked whether to save the entry to disk. Enter "Y" to save your changes, "N" to abandon them. You may also abandon a revision at any point by pressing <ESC>. If you elect to save the entry, it will be written to disk, then the current page is redisplayed, reflecting the changes you have just made. If you do not save the entry to disk, all your changes will be lost.

Revising the Modem Command

The modem dialing command, and its suffix, are settable in the Modem SetUp section of the SetUp command.

Adding or Revising Long Distance Codes

Long distance codes are used primarily for accessing alternate long-distance services. Four such codes are provided, identified by the characters "-", "+", "@" and "#". Enter "P" at the dialing directory "==" prompt to display a window containing the current long distance code definitions. Press any key to remove this window from the screen.

To revise a long distance code, enter the "R" command. Now enter the long distance code id character at the "Entry to revise" prompt. The current string for that long distance code will be displayed, and you will be asked for the new value. Control characters and carriage returns may be included in long distance codes by using translation conventions described earlier. Press <CR> to complete the long distance code revision. Pressing <CR> as the first character in the new string deletes that long distance code. If you do not save the changes to disk, they will be in effect for the current session only. Press <ESC> to abandon long distance code changes completely.

Long distance codes are used to provide access to alternate long distance carriers, to dial through office PBX or switchboard equipment, or for similar uses. Long distance codes may precede or follow the number in the dialing directory entry. For example, say you wish to call some long distance number using your Sprint account. Begin by setting up a long distance code. Choose "R" to revise, then select the ld code to change (We'll use "#"). Set the "#" ld code to "123-4567,,99999," where "123-4567" is your local Sprint telephone number, and "99999" is your Sprint account ID (or password). To dial an entry, say number 14, using Sprint, enter "#14" at the "==" prompt. This is what happens: the modem will dial the local Sprint number, wait six seconds (the comma is a Hayes modem command convention for a two second delay) then enter your Sprint access code. It then waits another 2 seconds (the final comma) then dials whatever happens to be specified in entry number 14. Similar tasks can be performed using different ld codes. And remember, each code may be placed either before (i.e. "#14"), after (i.e. "14#") or both before and after the entry to be dialed.

Deleting Entries

You can use the 'D' command to delete entries from your dialing directory. A window will open and prompt you for a range of entries to delete. Enter the starting and ending entry numbers for the range you

wish to delete. Leave the second field blank to delete a single entry.

) For example, to delete entries 15-25, enter "15" in the first space and "25" in the second. After verification, those 11 entries will be deleted. To delete just entry 37, enter "37" in the first space, and press <CR> in the second. You'll be asked to verify that you actually want the entries deleted before any action is taken.

Making a Call

To dial an entry in your dialing directory, simply type the entry number (1-100) at the "==" prompt. To dial with a long distance code, place the code's identifier before and/or after the entry number. For example, enter "14" to dial entry number 14. Enter "#14" to dial entry number 14 preceded by the long distance code identified by "#". Enter "14+" to dial entry number 14 followed by the '+' long distance code. ProComm will send the modem command, an optional long distance prefix, the number, an optional long distance code, and finally the modem command suffix to the modem. All five parts of the dialing string are translated using the conventions described earlier under string translation (see Section 2). The parameters for the dialed entry become the current settings and remain after the call is complete.

If a command file is linked to the entry being dialed, the redial facility will be used to place the call. When a connection is made, control is passed to the linked command file. Command files can be very useful for setting up custom terminal configurations, performing automatic logons, loading specific keyboard macro files and many other functions.

Manual Dialing

To dial a number not in your directory, enter the manual dialing command ("M") at the "==" prompt. Then enter the telephone number you wish to dial. ProComm will send the dialing command plus the string you entered, plus the modem command suffix. A long distance code indicator may be used if it is the first and/or last character in the string (that is, you can enter "+212-555-1552", "212-555-1552#" or "+212-555-1552#").

Printing the Directory

ProComm lets you output your directory to a printer, disk file or any DOS device. Type "L" at the "==" prompt, then specify where to send the file. Simply press <CR> to use the default of PRN. You can even output the directory to the COM port you are using and send a listing of your directory to the remote computer.

Automatic Redial with Circular Dialing Queue

) Some remote systems can be very hard to reach. ProComm makes it easier to access hard-to-reach systems with its automatic redial feature. Press Alt-R to invoke the redial procedure. When the redial window opens, specify one or more dialing directory entries separated by blanks, commas or semicolons. Entries may contain long distance codes. ProComm will

continuously redial the numbers in the list until you are connected. Pressing <CR> without specifying any entries will cause ProComm to dial the numbers that were in the list the last time it was used.

If the redial time is exceeded, or ProComm senses one of the modem's no connect strings (specified in the SetUp screen), the program will automatically move to the next number in the list. As each number is reached, it is removed from the circular queue. To remove an entry that is being called from the list without first connecting, press the Del key while that number is being dialed. You can also press the Space key to abort the current call and proceed to the next entry in the list.

Set the length of time ProComm waits for some response by pressing the End key; specify the new time to wait (from 5 to 100 seconds) and press <CR>. If you want this change to become permanent, respond "Y" to the "save to disk?" prompt. Be certain, however, that the your modem's delay timeout is set at least as high as the wait time you specify here; otherwise, the modem will timeout and disconnect before the redial wait time is reached. You can specify the redial timeout delay in the SetUp (Alt-S) screen as well as at the redial function.

Between calls ProComm will delay for the amount of time specified by the modem pause delay parameter (see Section 3). This delay is provided to allow the modem sufficient time to reset between calls. To continue immediately with the next call, press the Space key.

The alarm will sound when you connect during a redial. If a command file is linked to the entry that has connected, program control will pass to the linked command file. Command returns to you when that command file has finished its tasks.

6. FILE TRANSFER

The ability to transfer information makes a communications program a very powerful tool. ProComm gives you several choices for two-way information transfer via file transfer protocols. With them you can upload (send) and download (receive) files from almost any system.

Uploading Files

Begin a file upload to another system by starting the transfer on the remote computer. When the remote indicates that it is ready, press PgUp. You will be presented with a menu of protocol choices:

```
+-----+ UPLOAD +-----+
|
| 1)  XMODEM          |
| 2)  Kermit          |
| 3)  Telink          |
| 4)  MODEM7          |
| 5)  YMODEM          |
| 6)  YMODEM Batch    |
| 7)  ASCII           |
| 8)  COMPUERVE B     |
| 9)  WXMODEM         |
| ESC to Abort        |
|
| Protocol:           |
+-----+
```

Enter the number of the protocol you wish to use. A second window will prompt you for the name of the file to upload. Enter the filename, including a path if you wish, then press <CR>. Another window will open and report on the progress of the transfer.

An ASCII upload is something of a special case. No transfer progress window will be displayed, but the status line will indicate that an ASCII transfer is taking place. ASCII uploads are under the control of several setup options. The first set of options control pacing. You can set ProComm to pause after it sends each line of text or after each character to avoid overflowing the receiver. You may also set ProComm to wait to receive a specific character before sending the next line. Another option allows you to specify if text is to be echoed locally. In most cases you will want to let the remote do any echoing of transferred text. Still other options determine whether carriage returns or linefeeds should be stripped or translated from the outgoing file.

Downloading Files

Downloading files is just as easy. After starting the download procedure

on the remote, press PgDn. The protocol selection window is again displayed (but this time headed "DOWNLOAD"). Enter the number of the protocol you wish to use. If a Default Download Path has been specified in the General SetUp area, downloaded files will be placed in the specified directory. Otherwise they will be placed in the currently logged drive and directory.

If you select ASCII, XMODEM or YMODEM protocol, a window will open and you will be prompted for the filename. In all other cases the filename is provided by the sender and you do not need to enter it locally. If you enter the name of a file that already exists you will be asked if it should be overwritten. If a filename provided by the sender already exists, the received file will be renamed by placing a dollar sign in the first position of the filename. For example, if you select to download FILE1.EXE using the Telink protocol and that file already exists, the downloaded file will be named \$ILE1.EXE.

After you have specified the protocol and (possibly) the file name, a window will open which displays information regarding the transfer. When the transfer is complete, or an abort is sensed, the alarm will sound and appropriate messages will be displayed.

ASCII downloads are somewhat different from the others in two respects. First, no transfer status window is displayed. A message on the status line indicates that an ASCII download is in effect. Second, ASCII downloads need user intervention to complete. Text will continue to be captured until you press <ESC> to terminate the transfer.

There is, however, a large degree of control over various ASCII file transfer parameters. These can be set from the SetUp screen (Alt-S). You can conform CR and LF translation in a number of ways, allowing transfer to or from almost any system.

File Transfer Protocols

There are nine file transfer protocols available in ProComm:

ASCII

ASCII file transfer is the equivalent of typing information from one system to another. The ASCII characters are sent in a one-way stream with no handshaking (other than XON/XOFF, if it is activated) or error checking performed. This method is fine for some applications, but you are limited to text file transfer.

One difference between an ASCII download and file logging is that all data (including terminal escape sequences) is captured, while during logging terminal control sequences (with the exception of formfeeds) are filtered out.

ProComm provides a number of ASCII file transfer settings which allow you to transfer data to or from most any system. These options are settable via the ASCII Transfer SetUp section of the SetUp facility.

XMODEM File Transfer

XMODEM is a block-oriented error checking protocol released into the public domain by its creator, Ward Christensen. It is very popular on electronic bulletin board systems. XMODEM transfers only a single file at a time. The protocol uses two-way communications and either a checksum or cyclic redundancy check for error checking. XMODEM can handle text or executable files with over 99% accuracy. ProComm supports and automatically adjusts for both the checksum and CRC variants.

The XMODEM protocol is defined such that CRC checking is always attempted first. If CRC is not acknowledged by the sender then the checksum method is used. While ProComm correctly implements this system, however, some other systems do not. As a result, a problem can arise if the remote system responds to the CRC attempt but uses checksums.

Note that XMODEM requires transfers to be performed with 8 data bits, 1 stop bit and no parity. If you attempt to begin an XMODEM transfer while set to other parameters, ProComm will automatically switch to N/8/1, returning you to your original parameters when the transfer is complete.

Some systems, such as CompuServe, have trouble meeting the standard XMODEM timing sequences. You may set ProComm's XMODEM facility into a "relaxed" mode, which has less critical timing, using the SetUp (Alt-S) screen. Most other systems, however, will work fine with XMODEM set to normal.

MODEM7 File Transfers

MODEM7 is a variant of the XMODEM protocol. By sending the filename, batch transfers (multiple files) can be accomplished. CRC and checksum are supported.

YMODEM File Transfers

YMODEM is another XMODEM variant. Its main advantage is that it supports longer data blocks (1K) and thus speeds transfer times. ProComm supports YMODEM for single file transfers and YMODEM Batch for multiple file transfers. YMODEM Batch also provides some header information, such as filename and filesize. YMODEM file transfers are always performed with CRC error checking.

Telink File Transfers

Telink is yet another XMODEM/MODEM7 variant which adds file size and creation date information. It is found mainly on FIDO bulletin board systems, and provides for batch file transfers.

Kermit File Transfer

Kermit is a packet-oriented protocol developed at Columbia University and is available on many different computer systems. By using a technique called 8th-bit quoting, Kermit is able to transfer binary files between 7 and 8 bit systems. In some implementations, such as ProComm, Kermit

supports multiple file transfers.

ProComm's implementation of Kermit includes all of the latest Kermit enhancements, including data compression, file attributes, and sliding windows.

The most significant of these features is sliding windows. A "sliding window" protocol is a full duplex protocol that can transmit and receive data at the same time. The XMODEM family of protocols are half duplex protocols. They must wait between each block of data for a reply from the other side. XMODEM wastes quite a bit of time this way. Full duplex protocols can send a continuous stream of data while receiving replies at the same time, thus greatly increasing file transfer efficiency. ProComm Kermit will automatically sense if the other Kermit supports sliding windows and will use them if it does. Currently, Sliding Window Kermit is available on The Source, TCOMM BBS, and PC-HOST BBS. Mainframe versions are under development and should be appearing soon. ProComm Kermit is backward-compatible with earlier versions of Kermit.

ProComm's default Kermit settings are fairly standard and should need to be changed only under special conditions. Because of the sliding window enhancement, block size should be limited to a maximum of 90, rather than 94 as in standard Kermit.

In addition, a few Kermit server commands are supported. Press Alt-K to access the Kermit server command menu. Available commands include Finish, Logout, Send and Get. Get (option 1) and Send (option 2) will both prompt you for the appropriate filename and then perform the indicated function, i.e. either GET (receive) or SEND (transmit) a file. Finish (option 3) will terminate the Kermit session and return you to the system level. Logout (option 4) will terminate Kermit and log you off the remote. These commands are effective only when the remote Kermit is operating in server mode.

More information on Kermit is available from Columbia University. Send \$5 each for the Protocol or User manual to:

Kermit Distribution
Columbia University Center for Computing Activities
7th floor, Watson Laboratories
612 West 115th Street
New York, NY 10025

CompuServe B File Transfers

The CompuServe B protocol is available on the CompuServe Information Service. It may be used with ProComm in two ways. You may select it from the Upload or Download selection windows like any other protocol. You may also operate it in an automatic mode. To do so, set the ENQ parameter in the Terminal SetUp section to CIS B. This activates the automatic capabilities of ProComm to handle CompuServe B file transfers. All you need do at that point is instruct CIS to begin a transfer, and let the software do the rest. Be sure not to set ENQ to CIS B unless you are connecting to CompuServe or strange results may occur.

WXMODEM File Transfers

WXMODEM, another variant of the XMODEM protocol, is used primarily on the PeopleLink online service; it provides a sliding window protocol, similar to that discussed under "Kermit File Transfer", above.

7. COMMAND FILES

Command files are text files you create that contain ProComm commands. You can use command files to perform automatic logons, perform unattended file transfers and many other tasks. You can even link command files to the entries in your dialing directory so that the entire dialing and logon procedure is automated.

You can create a command file using virtually any word processor which saves in straight ASCII format; if your word processor normally makes use of special or extended ASCII characters, then you should use its "non-document" mode. A command file may have any valid filename; however, ProComm looks for files with an extension of .CMD when it lists available command files.

Command files can be executed from within ProComm or as a command line option when you first invoke the program. If you specify them on the command line, using the "/F" option, they will be executed immediately upon program startup. Execute them from inside the program by pressing Alt-F5. ProComm first searches the current directory for files with the .CMD extension. If any are found, their names are displayed in the window. If none are found, ProComm will look in the directory pointed to by the ProComm environment variable. Again, the names of any files found are displayed in the window. If no files are found in either directory, the message "NO FILES" will be displayed.

To execute a command file you may either type the filename or choose from the scrolling window display of available .CMD files. If you type the filename, you may omit the .CMD extension. To choose from the scrolling window, position the highlight on the name of the file you wish to execute and press <CR>. PgUp and PgDn will scroll the window one page in either direction. The arrow keys will scroll the window one entry in either direction. Pressing the Home key will position the list at the first page of available command files; pressing End will position it at the last. If you don't see the highlighted entry, use the Alt-Z command to change the color being used for highlighting so that it is different than that being used for regular foreground display. The command file window uses the colors displayed in Alt-Z for the file transfer window.

Once you begin to enter a command file name at the prompt you may still scroll the window, but you may not select a file for execution from the scrolling display.

Abort a command file by pressing <ESC> during its execution. The command file will abort when the current command has completed (which might take a few seconds). In some cases, such as while dialing or performing a file transfer, two <ESC>'s are needed: one to abort the task in progress and a second to abort the command file.

There is a special command file named PROFILE.CMD which ProComm looks for in the default directory first, then in the directory pointed to by the

ProComm environment variable (see Section 1). If PROFILE.CMD is found, it will be executed immediately upon program startup, before any other command file specified as a "/F" command line option. You can use the profile to do such custom setup work as changing to a special directory, turning off the sound, changing line settings, or overriding the default modem initialization string.

Command File Syntax

Command file commands begin with special command words, listed below. Commands may be entered in either upper- or lower-case. When processing commands ProComm looks only at the first 4 characters. Thus all the following commands are treated the same:

TRANSMIT = trans = Tran = TRANSM

Each command must appear on a separate line.

```
IF NOT CONNECTED           ; this is correct
    MESSAGE "sorry!"        ;
ENDIF                       ;
```

```
IF NOT CONNECTED MESSAGE "sorry!" ; this is incorrect
ENDIF                             ;
```

Many commands have one or more additional arguments; if the argument is listed in brackets ([]) it is optional, otherwise it is required. Arguments listed within quotation marks (" ") should include the quotation marks; thus the command RUN, if you wish to use it to run WordStar, would be typed

```
RUN "WORDSTAR"
```

Arguments may be separated by blanks or commas. Thus both

```
GETFILE XMODEM "FILE.EXT"
```

and

```
GETFILE,XMODEM,"FILE.EXT"
```

are valid.

To use the quotation character in a quoted string, precede it with the special escape character ` (the reverse tick mark, or accent grave, ASCII 96). Thus to print the message

```
She said "Goodbye" and then went home.
```

use the command

```
MESSAGE "She said `\"Goodbye`\" and then went home."
```

Labels are used as targets of GOTOs and GOSUBs. Labels must end with a colon. Below are some valid labels:

```
LABEL1:
```



```
This_is_a_long_label:  
JUMP1:  
split:  
GO_HERE:
```

Labels must appear on a line by themselves. (Comments are allowed on label lines; executable statements are not). Labels may be of any length; however, only the first 8 characters are used by the interpreter. Thus LABEL_TAG1: and LABEL_TAG2 are the same as far as ProComm is concerned.

Comments begin with a semi-colon (;). Any text following a semi-colon is treated as comment text. Below are valid comments:

```
; This is a comment.  
LABEL7:                ;This is a comment on a label line  
TRANSMIT "Welcome back" ;And this is a comment as well
```

There are 10 string variables, named S0-S9, which may be set and used in place of quoted strings. They have a maximum length of 80 characters each. They may be set by the ASSIGN, GET and RGET commands. String variables may be used in place of a quoted string in any of the commands marked below with the @ character. For example, the commands

```
ASSIGN S5 "Hello, Mike Todd here"  
TRANSMIT S5
```

and

```
TRANSMIT "Hello, Mike Todd here"
```

are functionally the same. One of the most useful applications of string variables is in obtaining and using user responses. Consider the commands below:

```
MESSAGE "Enter the name of the file to upload:"  
GET S4  
SENDFILE XMODEM S4
```

String variables are also a very handy method for passing values between command files. When you chain from command file to command file, using the EXECUTE command, string variable contents are not reset. Thus you may set a variable in one command file, and act on the variable in another.

ProComm allows the nesting of commands such as IF, SWITCH and GOSUB. Nesting, however, is limited to 10 levels.

Characters that are received from a remote system are stored in the receive buffer. During command file execution, the receive buffer is emptied before each command is executed, with the exceptions noted below. What this means is that before each command is performed, all the characters that have come in are displayed on the screen and are therefore not available to be processed by later commands. The exceptions are the GET, RGET and WAITFOR commands, as well as labels and comments. In these cases the buffer is not emptied, allowing the command to process the accumulated characters. The point here is that if you have a command sequence that looks like


```
TRANSMIT "password!"  
PAUSE 5  
SET DUPLEX HALF  
WAITFOR "target"
```

the text you are looking for may come in and be processed before the WAITFOR command has a chance to see it. A better solution would be to change duplex at some other point, and let the WAITFOR command do the extra pausing:

```
SET DUPLEX HALF  
TRANSMIT "password!"  
WAITFOR "target" 35 ; 30 seconds is the default pause
```

Use the commands described below to perform your specific task. Be sure to test your command files thoroughly before using them for unattended communications.

The following notations apply to the commands listed below: Commands marked with an asterisk (*) may be tested with the IF command. Commands marked with an at-sign (@) indicate where string variables may be used in place of quoted strings. Sx indicates that you should use one of the string variables. Ellipses (...) mean that you may place a number of command lines in that spot.

Top Level Commands

ALARM [seconds]

The ALARM command will sound an alarm to alert you to some event. Use the [seconds] option to determine the amount of time the alarm will sound. If the [seconds] option is not specified ProComm will use the Alarm Time specified in the General SetUp screen. This command is also under control of the Alarm Sound setting. Both Alarm Sound and Alarm Time may be regulated using the SET command described below.

Example: ALARM 5 ; sounds the alarm for 5 seconds

ASSIGN Sx "string" @

This command assigns the contents of "string" to a user variable. Use ASSIGN to set a user variable from within your command file.

Examples: ASSIGN S5 "12345" ; set S5 to contain the string '12345'
ASSIGN S6 S5 ; sets S6 to be the same as S5

BREAK [time]

The BREAK command is used to send a break to the remote system. The optional [time] argument determines the length of the break in milliseconds. If [time] is not specified, ProComm will use the default as indicated by the Break Length option in the Terminal SetUp screen.

Examples: BREAK ; send a break using the default timing
BREAK 500 ; send a 500 millisecond break

CHDIR "drive and/or directory" @

The CHDIR command will change the logged directory and/or drive.

Examples: CHDIR "A:" ; change the logged drive to A:
CHDIR "C:\COMM" ; change to \COMM dir on drive C:
CHDIR "\COMM" ; change logged dir to \COMM

CLEAR [bg fg]

The CLEAR command is used to clear the top 24 lines of your screen. The optional parameter [bg fg] (you must use both codes) allows you to change your background (bg) and foreground (fg) colors. If the [bg fg] option is not used, the screen will be cleared to the current colors. If the [bg fg] option is used, ProComm will clear the screen to the new colors as well as reset the current colors to those specified. The codes to use for colors are as follows:

| | | |
|-----------|---------------|------------------------------|
| 0 Black | 8 Dk Grey | (8-15 are the bright colors) |
| 1 Blue | 9 Lt Blue | |
| 2 Green | 10 Lt Green | |
| 3 Cyan | 11 Lt Cyan | |
| 4 Red | 12 Lt Red | |
| 5 Magenta | 13 Lt Magenta | |
| 6 Brown | 14 Yellow | |
| 7 Lt Grey | 15 White | |

Only the codes 0-7 are valid for background colors; any of the sixteen colors may be used for the foreground.

Examples: CLEAR 0 10 ; clear screen to lt green on black
CLEAR ; clear screen to current colors

DIAL "entry" @

The DIAL command is used to call an entry in your dialing directory. Specify the number of the entry, optionally preceded and/or followed by a long distance code identifier as the argument.

Examples: DIAL "5" ; call entry number 5
DIAL "#5" ; call entry 5 using ld code '#'

If a second command file is linked via the dialing directory to the entry being dialed, the linked command file will not be executed. Place all statements to be executed in the command file which initiates the call.

Use the IF LINKED command to avoid secondary dialing in command files linked to dialing directory entries. For example, including the commands

```
IF NOT LINKED          ; do not execute the dial command if
    DIAL "5"           ; this file is executing via linkage
ENDIF                  ; to a dialing directory entry
```

in a command file linked to entry number 5 allows you to use that command file both as a stand alone file and linked to the dialing directory entry.

This command uses the auto redial facility to place its calls; it will keep re-dialing until a connection is made. To make a call without using auto redial, use the TRANSMIT command.

```
Examples: TRANSMIT "ATDT1 314 449-9401!"      ; place the call
          PAUSE 10                            ; wait 10 seconds
          IF CONNECTED
              ...                             ; do these commands if
          ENDIF                               ; connected
```

Remember, if you sort your dialing directory you will need to change your command files so the entry numbers match.

DOS "command" [WAIT] @ *

The DOS command allows you to execute DOS commands or other programs from within a ProComm .CMD file. Enter the "command" as it would appear on the DOS command line. For example, to go out to DOS and type a file named FILE.EXT to your printer use the command:

```
DOS "type FILE.EXT > prn"
```

If the optional argument WAIT is included, ProComm will wait for a keystroke after executing the command, before returning to ProComm.

To execute this command properly you must make sure of two things. First, you must have enough memory to run the "command". Secondly, COMMAND.COM must be where ProComm can find it. (Either in the boot location or wherever COMSPEC is pointing). While this command returns an error status checkable with the IF command, the error check is very limited. The DOS command will indicate FAILURE only if COMMAND.COM was not found. IF COMMAND.COM was found, even if the command to be executed was invalid, SUCCESS will be returned.

```
Examples: DOS "del FILE.EXT"                  ; delete a file
          DOS "sortdisk"                      ; run a program called sortdisk
          IF FAILURE
              MESSAGE "COMMAND.COM not found"
          ENDIF
```

CAUTION: If you execute a program or command requiring user input be sure you are around to provide it, since the program will wait until you do.

EMULATE terminal -or- EMULATE "terminal"

The EMULATE command changes the active emulation to that specified. Valid terminal types are: VT100, VT52, IBM3101, TV920, TV950, ADM5, HEATH19, ANSI, ADDSVP and WYSE100.

Examples: EMULATE VT100 ; change emulation to VT100
 EMULATE "IBM3101" ; emulate the 3101

EXECUTE "cmd file" @

The EXECUTE command allows you to begin execution of a different command file. The currently executing command file will be ended and will not be returned to. The EXECUTE commands allows a one-way chaining of command file execution. The file extension of .CMD need not be specified.

Remember, string variables are not reset when chaining command files so you may use them to pass values. String variables are reset to null, however, when you begin the first command in the chain.

Examples: EXECUTE "CALL_KEN.CMD" ; execute CALL_KEN.CMD
 GET SO ; get the option
 SWITCH SO ; switch based on option
 CASE "A"
 ASSIGN S1 "CHOICEA.CMD"
 ENDCASE
 CASE "B"
 ASSIGN S1 "CHOICEB.CMD"
 ENDCASE
 DEFAULT
 ASSIGN S1 "DEFAULT.CMD"
 ENDCASE
 ENDSWITCH
 EXECUTE S1

EXIT

The EXIT command terminates the executing command file and returns you to Terminal Mode.

Examples: TRANSMIT "Goodbye" ; log off remote
 HANGUP ; hangup phone
 EXIT ; return to Terminal Mode

FIND Sx "target" @ *

The FIND command looks for an occurrence of the "target" string in the string variable Sx. Test for an occurrence of "target" within Sx using the IF FOUND command. The FIND command is not case sensitive.

Examples: MESSAGE "Enter the password:" ; prompt
 MGET SO ; get with mask
 FIND SO "password" ; look for password
 IF NOT FOUND ; found ?


```

        MESSAGE "Invalid password"      ; not found, do this code
        GOTO SECURITY_BREACH
ELSE
        GOSUB WELCOME                    ; found, do this
ENDIF

```

GET Sx [length]

The GET command is used to obtain and store user input. The Sx argument determines which string variable is used to hold the data. The optional [length] variable determines the maximum number of characters that will be accepted. If the [length] argument is not specified, the maximum size of 80 characters is used.

When responding to a GET command, the user must enter a <CR> to signal that his input is complete. The <CR> is not included in the string variable. If the [length] argument is used, the user will be allowed to enter up to [length] number of characters, but still must use a <CR> to complete his entry. ProComm will beep if the user attempts to enter more than [length] characters.

The MGET command is the same as the GET command except that the text the user enters is not displayed; rather each character typed will display as an asterisk (*). This is handy for security-related items such as passwords.

```

Examples: MESSAGE "Enter your choice: (A,B or C)"
          GET S3 1
          SWITCH S3
              ***
          ENDSWITCH

          MESSAGE "Enter the password"
          MGET S9 8
          FIND S9 "secret"
          IF NOT FOUND
              MESSAGE "You are not an authorized user."
              QUIT
          ENDIF

```

GETFILE

```

KERMIT                *
XMODEM "filename"     @ *
WXMODEM "filename"    @ *      (Widowed XMODEM)
RXMODEM "filename"    @ *      (Relaxed XMODEM)
YMODEM "filename"     @ *
BYMODEM               *      (YMODEM Batch)
TELINK                *
MODEM7                *
ASCII "filename"       @ *
CISB                   *      (CompuServe B)

```

The GETFILE command performs a file download (receive). A number of protocols are currently supported; see the section on file transfers for more information on each protocol.

To perform a download you must first initiate the transfer on the remote. When that system indicates that it is ready, begin your transfer.

Note that 4 protocols require you to specify the filename to receive; for the other protocols, the filename is provided by the sending system. All transfers may be checked for successful completion using the IF SUCCESS/FAILURE command.

```
Examples:  WAITFOR "Begin your transfer now" ; wait till it's ready
           GETFILE XMODEM "FILE.EXT"        ; receive FILE.EXT

           WAITFOR "Kermit-32>"             ; wait for prompt
           MESSAGE "Enter file to transfer" ; transfer a file with
GET S1                                ; Kermit
           TRANSMIT "SEND "                 ; send transfer command
           TRANSMIT S1                     ; the file name
           TRANSMIT "^M"                   ; and a CR
           GETFILE KERMIT                   ; now get it
```

GOSUB label

The GOSUB command provides for an unconditional branch with return. Upon encountering a GOSUB command, the interpreter searches the command file for the label specified. If the label is found, execution will continue with the command immediately following the label. If the label is not found, the command file will terminate with an "Unexpected end of file" error.

After successfully branching to the specified label, execution will continue until a RETURN command is found, at which point ProComm will jump back to the point at which the GOSUB was called. Execution resumes at the command immediately following the GOSUB. Each GOSUB must have its associated RETURN.

GOSUBs may be nested to a depth of 10 levels. If the end of the command file is encountered within a GOSUB, an "Unexpected end of file" error will occur. If you attempt to nest more than 10 GOSUB calls, a "Stack overflow" error will result. Likewise, if you attempt to RETURN without having a corresponding GOSUB, a "Stack underflow" error will occur.

```
Examples:  SWITCH S0 ; switch based on the contents of S0
           CASE "ABC"
             GOSUB LABEL1
           ENDCASE
           CASE "ZXY"
             GOSUB LABEL2
           ENDCASE
           DEFAULT
             GOSUB ERROR1
           ENDCASE
ENDSWITCH
...
```

; Subroutine area

```
LABEL1:
...
```



```
RETURN
```

```
LABEL2:
```

```
    ""  
RETURN
```

```
ERROR1:
```

```
    ""  
RETURN
```

GOTO label

The GOTO command performs an unconditional branch to the indicated label. Upon encountering a GOTO command, the interpreter searches the command file for the label specified. If the label is found, execution will continue with the command immediately following the label. If the label is not found, the command file will terminate with an "Unexpected end of file" error. Remember, only the first 8 characters of a label are actually used. If two identical labels exists, ProComm will branch to the one closest to the beginning of the file.

GOTOs may not be used to jump into the middle of IF or SWITCH statements, although they may be used to branch out of those constructs. They should also not be used to branch into or out of subroutines (code segments designed to be used with the GOSUB command), although they may be used within the boundaries of individual subroutines. Use of a GOTO in these situations will result in unexpected and usually erroneous execution.

```
Examples:  IF NOT WAITFOR  
            GOTO ERROR_EXIT          ; this is OK  
ENDIF  
  
    ""  
ERROR_EXIT:  
    MESSAGE "Abnormal termination"  
    HANGUP  
    QUIT  
  
GOTO LABEL1          ; this is not OK  
SWITCH SO  
    CASE "xyz"  
    LABEL1:  
    etc.
```

```
HANGUP      *
```

The HANGUP command attempts to disconnect the phone in the manner described for the Alt-H command in Section 3. Use the IF CONNECTED command to determine if you successfully disconnected.

```
Example:    HANGUP          ; disconnect the phone
```

HOST

The HOST command is used to put ProComm into Host Mode.

Example: HOST ; enter host mode

IF condition

The IF command is used to make decisions. The syntax of the IF statement is

```
IF condition
    [part 1]
ELSE
    [part 2]
ENDIF
```

where the ELSE part is optional. The condition is evaluated; if it is true, [part 1] is executed. If it is false, and there is an ELSE, [part 2] is executed. There must be an ENDIF for every IF. IF commands may be nested up to 10 levels deep.

Valid conditions for the IF command are

| | | |
|---------|-----------|---------|
| SUCCESS | CONNECTED | FOUND |
| FAILURE | LINKED | WAITFOR |

The SUCCESS condition is evaluated as true if the last checkable command was successfully executed. A "checkable" command is a command file command that sets one of the condition flags. Checkable commands are indicated in this chapter by having an asterisk (*) after the command name. For example:

```
RUN "someprog"
IF SUCCESS
    ...                ; this segment will execute if "someprog" ran.
ENDIF
```

The FAILURE condition is considered true if the last checkable command was not successfully completed. For example, if you had an RGET command that timed out, FAILURE would be considered true:

```
RGET S9 80 5
IF FAILURE
    ...                ; this segment will execute if RGET times out.
ENDIF
```

The CONNECTED condition is true if CD (Carrier Detect) is found to be high. CD is high when you are connected to a remote system, or if your modem is forcing the CD lead high. WARNING: be sure that your modem does not force CD high (usually a dip switch setting) or the CONNECTED condition will always be true.

```
IF CONNECTED
    ...                ; perform this segment if you are connected
ENDIF
```

The LINKED condition is considered true if the command file that is executing was started because it was linked to an entry in the dialing directory. In other words, if you have a command file called VAX.CMD which is linked to dialing directory entry number 5, and you call entry number 5 and are connected, and VAX.CMD begins execution, an IF command

that looks like

```
IF LINKED
    ...
ENDIF
```

will be considered true. The primary use for the LINKED condition is so you can write one command file and use it both stand alone and linked to dialing directory entries.

```
IF NOT LINKED
    DIAL 5
ENDIF
```

In this code segment, the DIAL command will be executed only if the command file was not executed because of a dialing directory link. (More on the NOT option below).

The FOUND condition is used to test the result of the last FIND command executed. It is considered true if the "target" was found in the specified string variable. For example, in the sequence

```
ASSIGN S9 "ABCDEFGH"
FIND S9 "CDE"
IF FOUND
    ...
ENDIF
```

the commands denoted by ellipses (...) would be executed, as the FOUND condition would be true.

The WAITFOR condition is used to check the result of the last WAITFOR command. If the "target" specified in the WAITFOR command was received, the WAITFOR condition would be true. If the WAITFOR command timed out before receiving the "target", the condition would be false.

```
WAITFOR "ABCDEFGH" 15      ; wait 15 seconds for 'ABCDEFGH'
IF WAITFOR
    ...                    ; execute if 'ABCDEFGH' was received
ELSE
    ...                    ; execute if timed out
ENDIF
```

The NOT operative may be employed with any of the conditionals. The effect of the NOT is to reverse the value of the condition. For example, if CONNECTED is false, then NOT CONNECTED would be true. The conditions NOT SUCCESS and FAILED are exactly the same. The code segments

```
IF FOUND                IF NOT FOUND
    [part 1]             [part 2]
ELSE                     ELSE
    [part 2]             [part 1]
ENDIF                   ENDIF
```

will result in identical execution.

```
Examples: IF NOT CONNECTED
            ...                ; do this if no connection
```



```

ENDIF

WAITFOR "Something"
IF WAITFOR
    ... ; do this if it was found
ELSE
    ... ; else do this
ENDIF

SEND_IT:
SENDFILE KERMIT "FILE.EXT"
IF NOT SUCCESS
    MESSAGE "Error in file transfer. Retrying..."
    GOTO SEND_IT
ENDIF

; How to redial without using Alt-R

DOIT:
TRANSMIT "ATDT123-4567"
WAITFOR "CONNECT" 20
IF NOT WAITFOR
    MESSAGE "No connect. Redialing..."
    GOTO DOIT
ENDIF

```

```
ISFILE "filename" @ *
```

The ISFILE command is used to determine if a specific file exists in the current directory. Use the IF SUCCESS/FAILURE command to test the results of the ISFILE command.

```

Examples: ISFILE "procomm.doc"
          IF SUCCESS
            MESSAGE "Doc file exists"
          ELSE
            MESSAGE "Doc file not found"
          ENDIF

          MESSAGE "Enter filename"
          GET SO
          ISFILE SO
          IF NOT SUCCESS
            MESSAGE "File does not exist"
          ENDIF

```

```

KERMSERVE
SENDFILE "filename" @ *
GETFILE "filename" @ *
FINISH
LOGOUT

```

The KERMSERVE command may be used to issue a Kermit server command. The available commands are listed above.

```
Examples: MESSAGE "File to send?" ; prompt for filename
```



```
GET SO                ; get filename
KERMSERVE SENDFILE SO ; send file
KERMSERVE FINISH      ; issue the FINISH server command
```

KFLUSH

The KFLUSH command is used to clear any accumulated keystrokes from the keyboard buffer. Any keystrokes that have been entered, but not processed, will be lost.

Examples: KFLUSH ; clear keyboard buffer

LOCATE row col

The LOCATE command positions the cursor to the location specified by row and col (column). Rows are numbered 0-24, columns 0-79, with 0,0 (row 0, col 0) being the upper left corner of the screen.

```
Examples: CLEAR          ; clear the screen
          LOCATE 10 20    ; position cursor
          MESSAGE "ENTER CHOICE:"
          LOCATE 10 44    ; position at end of line
          GET SB
```

LOG

```
OPEN ["filename"]      @ *
CLOSE
SUSPEND
RESUME
```

The LOG command controls file logging during command file execution. Use the OPEN command to start logging data to disk. If "filename" is not present, the Default Log File as specified in the General SetUp section will be used. Use the CLOSE command to turn off file logging. The SUSPEND command will stop text from being logged temporarily without closing the log file. Use the RESUME command to continue logging after a SUSPEND command.

```
Examples: LOG OPEN          ; use default log name
          ""
          LOG SUSPEND       ; put log on hold
          ""
          LOG RESUME        ; resume logging
          ""
          LOG CLOSE         ; close log
```

MACRO number -or- MACRO "number" @

The MACRO command will send the string currently assigned to any of the macro keys (Alt-0 through Alt-9). Use the MLOAD command to load individual keyboard macro definition files.

Examples: MACRO 5 ; send macro assigned to Alt-5


```
ASSIGN S8 "2"
MACRO S8 ; send macro assigned to Alt-2
```

```
MESSAGE "string" @
```

The MESSAGE command displays a string on the local console. The text is not sent to the remote. The message is displayed at the current cursor position, in the current colors. MESSAGE will always do a CR/LF after each string. The "string" may contain control characters such as CR and LF by using the translation conventions described in Section 2. Use the MESSAGE command for prompts, informational messages, building menus, etc.

```
Examples: MESSAGE "+-----+"
MESSAGE "! Enter your choice: !"
MESSAGE "+-----+"
LOCATE 2,20
GET S0 1
SWITCH S0
etc.
```

```
ASSIGN S9 "This is the first line^M^JThis is the second"
MESSAGE S9
```

```
MLOAD "filename" @
```

The MLOAD command is used to load a keyboard macro file.

```
Example: MLOAD "SYSTEM1.KEY" ; load a new macro file
```

```
PAUSE seconds -or- PAUSE "seconds"
```

The PAUSE command halts command file execution for the specified number of seconds. Characters received during a pause are not displayed until after the pause has completed.

```
Examples: TRANSMIT "Kermit send file.ext" ; start the transfer
PAUSE 3 ; let the remote start
KERMIT RECEIVE ; receive the file
```

```
PRINTER
ON
OFF
```

The PRINTER command is use to control print logging. Use the ON argument to begin logging the session to the printer; use OFF to end logging. ProComm writes the printer log to the DOS device PRN. You can use the MODE command in DOS to redirect printer output.

```
Examples: PRINTER ON ; begin print logging
PRINTER OFF ; end print logging
```

```
QUIT
```


The QUIT command terminates the executing command file and exits ProComm as well. Use it only to shut down the entire program.

```
Examples:  TRANSMIT "Logoff"      ; log off remote
           HANGUP                ; hangup the phone
           QUIT                  ; close down ProComm
```

RFLUSH

The RFLUSH command is used to clear the input buffer. Any characters that have been received, but not yet displayed, will be lost when this command is issued. It is generally used to clear the input buffer in preparation for some task.

```
Example:   RFLUSH                ; clear the input buffer
```

RGET Sx [length] [delay] *

The RGET command provides services similar to those of the GET command; however, input is taken from the remote computer rather than from the keyboard. The RGET command will complete when a <CR> is received or [length] characters have been received. Unlike with the GET command, a <CR> is not required if [length] characters have been received. Use the [delay] argument to specify the maximum number of seconds to wait for the string to be received before timing out. If [delay] seconds have elapsed without receiving a <CR> or [length] characters, the RGET command will timeout and execution will continue. You can determine if the command timed out using the IF SUCCESS/FAILED command. You must specify [length] if you wish to specify [delay]. If length is not specified, the maximum of 80 characters is used; if [delay] is not specified, the default of 30 seconds is used.

```
Examples:  TRANSMIT "ATSO=1^M"    ; go into auto answer
           WAIT_IT_OUT:
           IF NOT CONNECTED      ; wait for a connection
               GOTO WAIT_IT_OUT
           ENDIF
           TRANSMIT "ENTER PASSWORD:"
           RGET S9 8 45          ; wait 45 seconds max
           IF FAILED              ; timed out
               TRANSMIT "Times up. Goodbye"
               HANGUP
               GOTO WAIT_IT_OUT
           ENDIF
           FIND S9 "secret"
           IF NOT FOUND
               TRANSMIT "Sorry , but you're not authorized."
               HANGUP
               GOTO WAIT_IT_OUT
           ENDIF
```

RUN "program" [WAIT] @ *

The RUN command is similar to the DOS command except that it cannot

execute internal DOS commands. See the explanation of the DOS command for conditions necessary for the correct execution of this command. A major difference between the DOS and the RUN commands is the error codes returned. RUN will report any non-zero return code from the "program" as FAILURE; only "programs" exiting with a return code of zero will indicate SUCCESS.

If the optional argument WAIT is included, ProComm will wait for a keystroke after executing the program, before returning to ProComm.

Examples: RUN "filesort" ; execute a program called filesort

```

ASSIGN SO "filesort"
RUN SO
IF FAILURE
    MESSAGE "filesort returned error"
ELSE
    MESSAGE "filesort executed successfully"
ENDIF

```

SENDFILE

```

KERMIT "filename"          @ *
RXMODEM "filename"         @ *      (Relaxed XMODEM)
XMODEM "filename"          @ *
YMODEM "filename"          @ *
BYMODEM "filename"         @ *      (YMODEM Batch)
TELINK "filename"          @ *
MODEM7 "filename"          @ *
ASCII "filename"           @ *
CISB "filename"            @ *      (CompuServe B)
WXMODEM "filename"         @ *      (Windowed XMODEM)

```

The SENDFILE command performs a file upload (send). Many different protocols are currently supported; see the section on file transfers for more information on each protocol.

To perform an upload you must first initiate the transfer on the remote. When that system indicates that it is ready, begin your transfer.

All 9 protocols require that you specify the filename to send. All transfers may be checked for successful completion using the IF SUCCESS/FAILURE command.

```

Examples:  WAITFOR "Begin your transfer now" ; wait till it's ready
           SENDFILE XMODEM "FILE.EXT"       ; send FILE.EXT

           WAITFOR "Kermit-32>"             ; wait for prompt
           MESSAGE "Enter file to transfer" ; transfer a file with
           GET S1                            ; Kermit
           TRANSMIT "RECEIVE"                ; send transfer command
           TRANSMIT S1                      ; the file name
           TRANSMIT "^M"                    ; and a CR
           SENDFILE KERMIT S1                ; now send it

```

SET ...

The SET command is used to control various system parameters and options. It is fully described in the next section.

SNAPSHOT

The SNAPSHOT command performs a screen dump as described in Section 3.

Example: SNAPSHOT ; dump the screen contents to the .IMG file

SWITCH Sx

The SWITCH command provides special multi-way decision making. A SWITCH compares the value of a string variable against a number of constants and branches accordingly. The syntax of a SWITCH command is:

```

SWITCH S2                      ; switch based on S2
    CASE "target1"              ; if S2 = 'target1'
        ...                     ; do these commands
    ENDCASE                     ; until here
    CASE "target2"              ; any number of cases
        ...
    ENDCASE                     ; needed for each case
    DEFAULT                     ; if no previous case matches
        ...                     ; do these commands
    ENDCASE
ENDSWITCH                      ; ends the SWITCH

```

The string variable Sx contains some value, usually put there by a GET or RGET command. When a SWITCH command is found, ProComm begins looking for a CASE statement that contains a "target" which matches the string variable specified in the SWITCH. The match must be complete, although it is not case sensitive. Thus the values 'ABC', 'abc' or 'AbC' would all match a CASE 'abc' command, but the values 'ABCDE', 'XYZabc' or 'a b c' would not.

After finding a CASE that matches, ProComm will continue command file execution starting with the command immediately following the matching CASE. Execution continues until an ENDCASE command is found. At that point ProComm skips to the command immediately following the matching ENDSWITCH command and resumes execution. Note that each CASE statement must have a matching ENDCASE, and each SWITCH a matching ENDSWITCH.

There is a special case known as the DEFAULT case. The commands within the DEFAULT case will be executed if no previous CASE matched the string variable. The DEFAULT case is optional and need not be specified. If there is no DEFAULT case, and no other match is found, command file execution will continue with the statement following the ENDSWITCH command.

Another special case the _NULL case. The _NULL case will be executed if the string variable being switched on is null, i.e., has no value. For example, if a user enters a CR only in reply to a GET or RGET command, the string variable being gotten will be null. The _NULL case would then be triggered. The correct syntax for the _NULL case is:

```
CASE "_NULL"
```


Examples: (display some menu of options)

```

GET_CHOICE:
MESSAGE "Enter your choice"
GET S5
SWITCH S5                                ; switch based on S5
    CASE "A"                             ; if S5 = 'a' or 'A'
        GOSUB CHOICEA                    ; do this
    ENDCASE
    CASE "B"                             ; if S5 = 'b' or 'B'
        GOSUB CHOICEB                    ; do this
    ENDCASE
    CASE "_NULL"                         ; do this if user pressed
        GOSUB NULL_CASE                  ; only CR
    ENDCASE
    DEFAULT                             ; S5 not = to 'A' or 'B' or NULL
        MESSAGE "Invalid selection"
        PAUSE 3
        GOTO GET_CHOICE
    ENDCASE
ENDSWITCH

MESSAGE "Continue? (Yes/No)"             ; prompt
GET S7                                   ; get response
SWITCH S7                               ; switch based on response
    CASE "NO"                           ; they said "no"
        QUIT                            ; so leave
    ENDCASE
ENDSWITCH

```

One use of a SWITCH command is to get a choice from the user, and then to perform various tasks depending upon the option selected. In the first example, the command file would begin by presenting some list of options. You could easily build a menu using the MESSAGE and LOCATE commands. The command file could then prompt the user for his selection, and GET the selection into a string variable, in this case S5. The command file will branch based on the user's selection. If the user entered the letter 'A', the SWITCH will execute the first case and call the subroutine entitled CHOICEA. After returning from that subroutine, execution will continue with the first statement following the ENDSWITCH command. If the user had entered 'B' as his choice, the CHOICEB subroutine would have been executed.

If neither 'A' nor 'B' had been entered by the user the DEFAULT case would execute. In that event the "Invalid selection" message would be displayed, the program would pause for 3 seconds, and then branch back up to the GET_CHOICE label and the process would be repeated.

In the second example, if the user had entered 'No' the first case would execute and the command file would QUIT. In all other cases, no part of the SWITCH would execute, and the command file would continue.

TRACE ON/OFF

The TRACE command allows you to "trace" the execution of a command file. If you set TRACE ON, every command will be printed to the screen, in

contrasting colors, as it is executed. TRACE is very handy for debugging errant command files.

Examples: TRACE ON ; set trace on

TRANSMIT "string" @

The TRANSMIT command sends a character string to the remote. The "string" may contain control characters using the standard translation conventions.

```
Examples: MESSAGE "Enter your ID"      ; send the prompt
          GET S6                        ; get the ID
          TRANSMIT S6                  ; send the ID
          TRANSMIT "!"                 ; plus a CR

          WAITFOR "First name:"        ; Wait for a prompt
          TRANSMIT "TOM!"              ; send the name plus a CR
```

WAITFOR "target" [delay] @ *

The WAITFOR command allows you to wait for a particular string to be received from the remote before command file execution continues. The [delay] option tells ProComm how many seconds to wait for the "target" before timing out and continuing execution. If no [delay] is specified, ProComm will wait 30 seconds.

Use the IF WAITFOR condition to test the results of a WAITFOR command.

```
Examples: WAITFOR "first name:" 45      ; wait for the string "first name"
          ; for 45 seconds
          IF WAITFOR                    ; if it was found w/o timing out
            TRANSMIT "TOM!"             ; send your name
          ELSE                           ; else
            GOTO ERROR                  ; go to error processing
          ENDIF
```

WAITFOR targets are not case sensitive, thus either "FIRST NAME" or "First Name" would successfully complete the command sequence above. You may also include control character in the target using the translation conventions described earlier.

Example: WAITFOR "^M^JBUSY" ; wait for CR LF then BUSY

WHEN "target" "response" @

The WHEN command is used to transmit a certain "response" text every time a certain "target" text is received. Once a WHEN command is initiated, it will be in effect until a CWHEN (Clear WHEN) command is found or the command file ends. For example, suppose you are calling an online service that displays a "-more-" prompt at the end of each page of display, and waits for you to enter a carriage return before it continues. By specifying the command

```
WHEN "-more-" "^M"
```


at the beginning of the command file, you are instructing ProComm to transmit a CR (^M) every time it receives the string "-more-", thus relieving you of the task of matching every occurrence of "-more-" with a

```
WAITFOR "-more-"
TRANSMIT "^M"
```

command sequence.

```
Examples:  WHEN "continue? (Y/N)" "Y^M"      ; send a 'Y<CR>' for
          ...                                ; every "continue" received
          CWHEN                               ; turn off WHEN processing
```

Set Commands

All SET commands have the same format:

```
SET parameter value
```

The SET commands are used to change the value of the parameters in the SetUp facility (Alt-S) and the Line Settings window (Alt-P). Available values are separated by slashes (/). Because those parameters are described in detail in other sections, an in-depth discussion is not presented here. The appropriate SetUp section is indicated in parenthesis for each command.

```
SET ASCII etc.                                (ASCII Transfer Setup)
    The ASCII transfer SET commands are covered below.
```

```
SET ALARM ON/OFF                             (General Setup)
    Controls alarm sound.
```

```
SET ATIME seconds                            (General Setup)
    Sets amount of time alarm sounds.
```

```
SET BACKSPACE IN NONDEST/DEST                (Terminal Setup)
    Controls destructive nature of received BS.
```

```
SET BACKSPACE OUT BACKSPACE/DELETE           (Terminal Setup)
    Controls character sent when BS key pressed.
```

```
SET BAUDRATE 300/1200/4800/9600/19200        (Line Settings)
    Sets the baud rate.
```

```
SET BREAK milliseconds                       (Terminal Setup)
    Sets length, in milliseconds, of break condition.
```

```
SET CR_IN CR/CR_LF                           (Terminal Setup)
    Controls incoming CR translation.
```

```
SET CR_OUT CR/CR_LF                         (Terminal Setup)
    Controls outgoing CR translation.
```


SET DATABITS 7/8 (Line Settings)
Sets the data bits used.

SET DLDIR "path" @ (General Setup)
Sets the default download drive and directory.

SET DUPLEX HALF/FULL (General Setup)
Controls the duplex setting.

SET ENQ ON/OFF/CISB (Terminal Setup)
Controls response to ENQ (ASCII 5).

SET FLOWCTRL ON/OFF (Terminal Setup)
Controls use of XON/OFF flow control.

SET HOSTPSWD "string" @ (Host Mode Setup)
Sets the host mode password.

SET PARITY EVEN/ODD/NONE/MARK/SPACE (Line Settings)
Controls parity.

SET PORT COM1/COM2/COM3/COM4 (Line Settings)
Selects serial port to use.

SET RDELAY seconds (Modem Setup)
Determines timeout period for redials.

SET SCROLL ON/OFF (Terminal Setup)
Determines scroll setting.

SET SHELLPSWD "string" @ (Host Mode Setup)
Sets the host mode shell password.

SET SOUND ON/OFF (General Setup)
Controls sound effects.

SET STOPBITS 1/2 (Line Settings)
Sets the stop bits used.

SET SWRITE BIOS/DIRECT (General Setup)
Determines the screen write method.

SET TRANSLATE ON/OFF (General Setup)
Controls use of the translate table.

SET TXPACE millisecs
Determines pacing for all outgoing character strings.

SET WRAP ON/OFF (Terminal Setup)
Controls line wrap.

Set ASCII Commands (ASCII Transfer Setup)

SET ASCII BLANKEX ON/OFF
Controls expansion of blank lines during ASCII uploads.

SET ASCII CHARPACE milliseconds
Sets the character pacing (in milliseconds).

SET ASCII DN_CR CR/CR_LF/STRIP
Controls translation of incoming CRs during ASCII downloads.

SET ASCII DN_LF LF/CR_LF/STRIP
Controls translation of incoming LFs during ASCII downloads.

SET ASCII ECHO ON/OFF
Controls local echo during ASCII uploads.

SET ASCII LINEPACE tenths
Sets line pacing timing (in 1/10 seconds).

SET ASCII PACECHAR number
Sets the pace character used. Specify as an ASCII decimal value.

SET ASCII UP_CR CR/CR_LF/STRIP
Controls translation of outgoing CRs in ASCII uploads.

SET ASCII UP_LF LF/CR_LF/STRIP
Controls translation of outgoing LFs in ASCII uploads.

Set Kermit Commands

SET KERMIT CQUOTE char
Set the Ctrl quote character.

SET KERMIT PACKSIZE number
Set the maximum packet size.

SET KERMIT PADCHAR char
Select the pad character.

SET KERMIT PADNUM number
Set the number of pad characters.

SET KERMIT BQUOTE char
Select the 8th bit quote character.

SET KERMIT HANDSHAKE char
Select the handshake character.

SET KERMIT EOLCHAR char
Select the end of line character.

SET KERMIT BLOCKCHECK 1/2/3
Select the block check type. 1 = 1 byte checksum, 2 = 2 byte checksum, 3 = 3 byte CRC.

SET KERMIT FILETYPE TEXT/BINARY
Select the transfer file type.

Arguments listed as "char" should use the ASCII decimal value of the desired character. For example, to use XON (ASCII 17) as the HANDSHAKE

character, issue the command

SET KERMIT HANDSHAKE 17

ERROR MESSAGES

If an error is encountered during the execution of a command file, an error message will be displayed and command file execution will stop. Error messages begin with the error number (described below), followed by the line number of the line containing the error, followed by the erroneous line itself, for example:

ERROR 2, LINE 6: goto missing_label

| ERROR NUMBER | DESCRIPTION |
|--------------|---------------------------------|
| 1 | Invalid token. |
| 2 | Unexpected end of file. |
| 3 | Unexpected CASE statement. |
| 4 | Unexpected DEFAULT statement. |
| 5 | Unexpected ENDCASE statement. |
| 6 | Unexpected ENDSWITCH statement. |
| 7 | Unexpected ELSE statement. |
| 8 | Unexpected ENDIF statement. |
| 9 | Unexpected token. |
| 10 | Missing token. |
| 11 | Stack overflow. |
| 12 | Stack underflow. |

ERROR 1 - Invalid token

Indicates that an unidentified keyword was found, or that an invalid argument was included.

ERROR 2 - Unexpected end of file

Indicates that the end of file was reached while executing some command. May be triggered by IF statements without their corresponding ENDIF; SWITCH statements without an ENDSWITCH; or when attempting to find a label that does not exist.

ERROR 3 to 8 - Unexpected [token]

Indicates that the given keyword was found in an unexpected place, i.e. an ELSE command with no preceding IF, or a DEFAULT command with no preceding SWITCH.

ERROR 9 - Unexpected token

Indicates that a keyword or argument was supplied where none are expected.

ERROR 10 - Missing token

Indicates that a token or argument was expected, but not supplied.

ERROR 11 - Stack overflow

Indicates that nesting has gone too deep, i.e. an 11th level in a nested if statement, or an attempt to access an 11th level subroutine.

ERROR 12 - Stack underflow

Indicates the attempt to return a level when no nesting has occurred. For example a RETURN command when no GOSUB has executed.

8. H O S T M O D E

ProComm includes a limited remote access facility known as Host Mode. Host Mode allows remote access to your computer for tasks such as file transfer and DOS shell access. In addition, it provides password security, a logon message, and can maintain a history of logons. Host Mode is intended to allow a user access to his home machine from work (or vice versa); it is not intended as a multi-user message base or bulletin board system.

Host Mode Setup

Host mode requires some setup before it can be used. Setup is composed of three basic parts: modem, operating system, and ProComm. Each must be correctly configured before attempting to use Host Mode.

Modem Setup

Host Mode depends quite a bit on the correct installation of your modem. Because there are many different modems available, not all of which we are familiar with, you may have to do a bit of reading in your modem manual.

First, Carrier Detect (CD) on your modem must be set to follow the true state of carrier. Carrier detect must not be forced true, or high, by dip switch settings. Carrier detect also should not be set to follow DTR. It should be set to follow the "real state", or the "RS-232 convention" or however your manual says it. Carrier detect is usually controlled by a dip switch setting on your modem, although on some new modems, like the Hayes 2400, you use a software command like &C1 and &D2. On a Hayes 1200 external modem you should have dip switch 6 in the UP position.

Next, data terminal ready (DTR) should also follow the real state. It should not be forced high. On a Hayes modem, this means switch 1 is UP.

Your cable might also have an impact on correct operation. Your modem manual should have an explanation of proper cable configurations.

Operating System Setup

Operating system setup revolves around the CONFIG.SYS file. CONFIG.SYS is a configuration file which contains commands used to configure your system. Each time you start DOS, the operating systems searches the root directory of the drive you booted from for a file called CONFIG.SYS. If it is found, DOS reads the file and interprets the commands it contains.

You may create a CONFIG.SYS file using your favorite editor or word processor (be sure to save it as a non-document, or ASCII file). If you add or change any of the commands in the configuration file, or create a new one, the changes are not in effect until the next time you start DOS, so be sure to reboot. Consult your DOS manual for more information about configuring your system.

The command in the CONFIG.SYS file that we are concerned with is the FILES= statement. This command controls the number of files that can be open at one time. The operating system default is 8; for ProComm we recommend 20 or higher.

To set this statement, create (or edit) a CONFIG.SYS file in the root directory of the disk that you boot with. Include in the CONFIG.SYS file a line that says

```
FILES=x
```

where x is 20 or greater. A quick and easy way to create the CONFIG.SYS file is this:

At the DOS prompt type

```
COPY CON CONFIG.SYS <CR>
FILES=20 <CR>
^Z <CR>                                (Press Ctrl-Z then <CR>)
```

Remember, the CONFIG.SYS file, containing the FILES= statement, must be present in the root directory when you boot. It has no effect otherwise. Increasing the FILES= parameter can also alleviate other problems indicated by "CAN'T OPEN FILE" messages and similar notices.

ProComm Setup

The SetUp facility (Alt-S) contains a section for Host Mode Setup. The Host Mode setup screen contains six host options which must be initialized. These options are also described in Section 3.

The first option is the Host ID String. The Host ID String is a text message that is displayed to callers after they connect. It is usually used as a greeting.

The second option is the modem auto-answer string. This string is the command sent to the modem to place it into auto answer mode. The default string, set for Hayes compatible modems, is "~~~+++~~~ATSO=1!", where

~~~ is a 1 1/2 second pause

+++ drops the modem into command state

~~~ is a 1 1/2 second pause

AT is the command prefix

SO=1 sets modem to answer after one ring

! sends a <CR>.

When host mode is exited, the Modem Initialization String (specified in the Modem setup section) is sent to the modem to reset it. If you do not want auto-answer to be on when you are not in host mode, include the command to turn it off within the Modem Initialization String. That command is S0=0 for Hayes compatible modems.

The third host option is the Host Access Password. It may be up to 8 characters long. Callers must match the password exactly, including upper- and lower-case. If you set the host access password to null, callers need only press <CR> at the password prompt to be allowed on your system.

The fourth host option is the DOS Shell Password. This password provides a second level of protection before callers are allowed access to the system level. Be very careful with this one; you don't want just anyone to have access to the operating system level on your machine.

The fifth host setup option is the Auto Baud Detect method. Three choices are available: NONE, KEY HIT and MODEM MSG. These choices determine what technique, if any, ProComm will use to attempt to synchronize baud rates with callers.

The last host option is the Connection Type. This allows you to run Host Mode either directly connected to another machine, or by using a modem. When set to Direct, ProComm automatically assumes a connection exists, and does not look for carrier or attempt baud rate synchronization.

Once you have set the options and assured that your modem is correctly configured, you can place ProComm into Host Mode by pressing Alt-Q. The auto-answer string is sent to the modem, and the local console displays "Waiting...". In addition, a line is added to PROCOMM.HST, the host audit trail history file, that records the time and date.

When a call comes in, and an Auto Baud Detect method other than None is specified, ProComm will attempt to match baud rates. After the connection is complete, ProComm will send the Host ID String and prompt the user for his name. The name is for informational purposes only. The caller's name is then recorded in the history file. ProComm then prompts the caller for the password. The caller must match the password completely, including case. All password attempts, both successful and otherwise, are recorded in the history file. Callers get three chances to match the password; if they are unsuccessful after three tries, ProComm will hang up on them.

If the caller makes it past the password, ProComm will display the contents of a special file called PROCOMM.MSG, if that file exists. You can use this file as a welcome message, to display character graphics, or whatever. Every 23 lines ProComm will halt, display a "[MORE]" message, and wait for any keystroke from the caller. When a keystroke is received, the display of the file continues. Use of the .MSG file is optional.

After PROCOMM.MSG is displayed (if it is), the caller is presented with the host menu:

F)iles C)hat D)ownload U)pload S)hell G)oodbye

Callers make their choices by entering the first letter of an option and pressing <CR>.

The F)iles option will send a list of the files in the current directory to the caller. The list is not displayed on the local console. F)iles uses the same facility as the Alt-F command, described earlier. If the caller chooses C)hat, an alarm on the host computer will sound for 30 seconds. The Host operator can press Space to chat with the caller. Pressing any other key will end the alarm and redisplay the host menu. If you have the alarm sound set off (via general setup at the Alt-S menu) no bells will sound on the local end. Press <ESC> to return to the host menu when you are done chatting.

D)ownload and U)pload work pretty much the same. The caller is presented with the file transfer menu:

K)ermit M)odem7 T)elink X)modem Y)modem batch

After selecting a protocol the caller may be prompted for a filename, depending upon the transfer direction and protocol used. The standard file transfer routines are used to perform the transfers. Users are only allowed to download files in the current directory. Similarly, all files uploaded will go to the current directory.

The S)hell command is very powerful, but it also has the potential of being very dangerous. It performs the equivalent of a "CTTY COMx" command, which basically redirects all console I/O out the COM port. The Shell command puts the caller in charge of your computer at the operating system (DOS) level. This means that a user can format your disk or whatever, so be very, very careful about who is allowed access to this command. Use the Shell Password to protect yourself. To return to ProComm, the user must enter "EXIT" at the DOS prompt.

There are some restrictions when using the S)hell command. For one, do not run programs that write directly to the screen buffer. This will cause the host machine to appear to lock up. If the host machine is running Dosedit, CED or a similar program, backspaces, escapes and possibly other key sequences can lock up the machine. This problem arises from DOS and the other programs, not ProComm: there is nothing we can do about it. In addition, some machines we have tried the host mode on do not pipe I/O to the correct port. We have not yet determined if this is a DOS or BIOS problem, but be aware that it may occur. We suggest that you thoroughly test Host Mode before making it available to callers.

The G)oodbye option logs the caller out and places a notice to that effect in the history file.

One additional command that is not listed on the menu allows a caller to shut down Host Mode from the remote side. If a user enters a Ctrl-Z, he will be prompted for a password. ProComm uses the Shell Password for this feature. If the user correctly enters the password, he will be logged off and ProComm will drop out of Host Mode. This is useful if you begin Host Mode from a command file, and wish to continue with the execution of that command file.

On the local side, the host operator has several options. You may press <ESC> to leave host mode and return to Terminal Mode. Press Ctrl-X to

log off the current user. All other keystrokes act as if the caller sent them and thus allow you to help novice users through the menus.

NOTE: Host mode, especially the S)hell command, opens a few more files. You must be certain that the FILES= statement in your CONFIG.SYS file is set high enough, preferably at FILES=20.

APPENDIX A - TERMINAL EMULATION

Overview

ProComm emulates a number of popular terminals. ProComm handles most of the standard features and many of the extended features of the terminals emulated. Functions of these terminals that are not emulated are still processed to insure that all control codes are handled properly.

Because of the wide variety of protocol convertors on mainframe front-end processors, some keys may not function as expected. These emulations have been tested with the mainframe systems at the University of Missouri, Columbia and the University of California, Berkeley, as well as within private industry. If you encounter a problem, refer to the TRANSMITTED CODES column in the terminal emulation charts on the following pages to see if ProComm is sending the code your protocol convertor is expecting.

If you encounter any errors in these emulations or have any questions, please let us know.

Digital Equipment Corporation VT-100 and VT-102

ProComm supports the standard VT-100 and VT-102 functions. Supported functions include: full/half duplex, set/reset modes, scroll region, special graphics character set, US & UK character sets, keypad application mode, cursor control, erase functions, insert/delete lines, full display attributes (including extensions for ANSI color graphics), programmable tabs and printer control functions. 132 column mode is not supported. ProComm responds to the Identify and Device Attributes commands with ESC[?1;2c (VT-100 with advanced video option). ProComm responds to the Ctrl-E (ENQ) enquiry function by sending the string stored for keyboard macro Alt-O. This "answerback message" is sent only if ANSWER BACK is enabled in ProComm setup (Alt-S). Keypad Application Mode functions are mapped to ProComm function keys as shown in the chart below.

| ProComm KEYS | DEC VT-100 FUNCTION | TRANSMITTED CODES |
|------------------|-------------------------------------|-------------------|
| Tab | Horizontal Tab | 0x09 |
| Delete | Character Delete | 0x7F |
| Home | Home Cursor | 0x1B5B48 |
| Cursor Up | Cursor Up | 0x1B5B41 |
| Cursor Down | Cursor Down | 0x1B5B42 |
| Cursor Left | Cursor Left | 0x1B5B44 |
| Cursor Right ... | Cursor Right | 0x1B5B43 |
| Ctrl-PgDn | Clear screen | 0x1B5B481B5B324A |
| End | Erase end of line | 0x1B5B4B |
| Ctrl-Home | Insert Line | 0x1B5B4C |
| Ctrl-PgUp | Delete Line | 0x1B5B4D |
| Ctrl-J | Line Feed | 0x0A |
| F7 | Keypad Application mode 1 | 0x1B4F71 |
| F8 | Keypad Application mode 2 | 0x1B4F72 |
| Shift-F7 | Keypad Application mode 3 | 0x1B4F73 |
| F5 | Keypad Application mode 4 | 0x1B4F74 |
| Shift-F6 | Keypad Application mode 5 | 0x1B4F75 |
| Shift-F5 | Keypad Application mode 6 | 0x1B4F76 |
| F3 | Keypad Application mode 7 | 0x1B4F77 |
| F4 | Keypad Application mode 8 | 0x1B4F78 |
| Shift-F3 | Keypad Application mode 9 | 0x1B4F79 |
| F9 or F10..... | Keypad Application mode 0 | 0x1B4F70 |
| F1 | Program Function 1 (PF1) | 0x1B4F50 |
| F2 | Program Function 2 (PF2) | 0x1B4F51 |
| Shift-F1 | Program Function 3 (PF3) | 0x1B4F52 |
| Shift-F2 | Program Function 4 (PF4) | 0x1B4F53 |
| Shift-F4 | Keypad Application mode DASH | 0x1B4F6D |
| Shift-F5 | Keypad Application mode COMMA | 0x1B4F6C |
| Shift-F9 | Keypad Application mode PERIOD ... | 0x1B4F6E |
| Shift-F8 | Keypad Application mode ENTER | 0x1B4F4D |

Mapping of VT-100 Keypad Application Mode Functions

PROCOMM FUNCTION KEYS

| F1 - F10 | | SF1 - SF10 | |
|----------|-----|------------|-----|
| PF1 | PF2 | PF1 | PF2 |
| 7 | 8 | 9 | - |
| 4 | 5 | 6 | , |
| 1 | 2 | 3 | E |
| | | | N |
| | | | T |
| 0 | . | | R |

DEC VT100 KEYPAD

| | | | |
|-----|-----|-----|-----|
| PF1 | PF2 | PF3 | PF4 |
| 7 | 8 | 9 | - |
| 4 | 5 | 6 | , |
| 1 | 2 | 3 | E |
| | | | N |
| | | | T |
| 0 | . | | R |

Keypad Application Mode for VAX/VMS EDT Editor

PROCOMM FUNCTION KEYS

| F1 - F10 | |
|------------|-----------|
| GOLD | HELP |
| PAGE | SECT |
| [CMD] | [FILL] |
| ADVANCE | BACKUP |
| [BOTTOM] | [TOP] |
| WORD | EOL |
| [CC] | [DEL EOL] |
| LINE | |
| [OPENLINE] | |

| SF1 - SF10 | |
|------------|---------|
| FIND NEXT | DEL L |
| [FIND] | [UND L] |
| APPEND | DEL W |
| [REPL] | [UND W] |
| CUT | DEL C |
| [PASTE] | [UND C] |
| CHAR | |
| [SPECINS] | ENTER |
| SELECT | [SUBS] |
| [RESET] | |

Press GOLD get first to get bracketed [] functions

IBM 3101

ProComm supports the standard IBM 3101 Model 1x/2x functions. Block mode is not supported. Supported functions include: full/half duplex, full character set, scroll on/off, program function keys, cursor control, and erase functions.

| ProComm KEYS | IBM 3101 FUNCTION | TRANSMITTED CODES |
|--------------------|--------------------------------|-------------------|
| Tab | Horizontal Tab | 0x09 |
| Delete | Character Delete | 0x7F |
| Home | Home Cursor | 0x1B48 |
| Cursor Up | Cursor Up | 0x1B41 |
| Cursor Down | Cursor Down | 0x1B42 |
| Cursor Left | Cursor Left | 0x1B44 |
| Cursor Right | Cursor Right | 0x1B43 |
| Ctrl-PgDn | Clear screen | 0x1B4C |
| End | Erase end of line | 0x1B49 |
| Ctrl-End | Erase end of screen | 0x1B4A |
| F1 | Program Function 1 (PF1) | 0x1B610D |
| F2 | Program Function 2 (PF2) | 0x1B620D |
| F3 | Program Function 3 (PF3) | 0x1B630D |
| F4 | Program Function 4 (PF4) | 0x1B640D |
| F5 | Program Function 5 (PF5) | 0x1B650D |
| F6 | Program Function 6 (PF6) | 0x1B660D |
| F7 | Program Function 7 (PF7) | 0x1B670D |
| F8 | Program Function 8 (PF8) | 0x1B680D |

Televideo 900 Series

ProComm supports the standard Televideo 900 series functions. Supported functions include: full/half duplex, program function keys, cursor control, erase functions, and full display attributes. The main difference between the 910/920 and the 925/950 emulations is in the codes generated for cursor down and in control codes for display attributes. The special graphics character set and user loadable status line are supported.

| ProComm KEYS | TELEVIDEO 9xx FUNCTION | TRANSMITTED CODES |
|--------------------|------------------------------|-------------------|
| Tab | Horizontal Tab | 0x09 |
| Backtab | Reverse Tab | 0x1B49 |
| Insert | Insert Character | 0x1B51 |
| Ctrl-Home | Insert Line | 0x1B45 |
| Delete | Delete Character | 0x1B57 |
| Ctrl-PgUp | Delete Line | 0x1B52 |
| Home | Home Cursor | 0x1E |
| Cursor Up | Cursor Up | 0x1B |
| Cursor Down | Cursor Down (910/920) | 0x0A |
| | Cursor Down (925/950) | 0x16 |
| Cursor Left | Cursor Left | 0x18 |
| Cursor Right | Cursor Right | 0x1C |
| Ctrl-PgDn | Clear Screen | 0x1A |
| End | Line Erase | 0x1B54 |
| Ctrl-End | Page Erase | 0x1B59 |
| F1 | Function 1 (F1) | 0x01400D |
| F2 | Function 2 (F2) | 0x01410D |
| F3 | Function 3 (F3) | 0x01420D |
| F4 | Function 4 (F4) | 0x01430D |
| F5 | Function 5 (F5) | 0x01440D |
| F6 | Function 6 (F6) | 0x01450D |
| F7 | Function 7 (F7) | 0x01460D |
| F8 | Function 8 (F8) | 0x01470D |
| F9 | Function 9 (F9) | 0x01480D |
| F10 | Function 10 (F10) | 0x01490D |
| Shift-F1 | Function 11 (F11) | 0x014A0D |
| Shift-F2 | FUNCT | *see below |
| Shift-F3 | Shift Line Erase | 0x1B74 |
| Shift-F4 | Shift Page Erase | 0x1B79 |
| Shift-F5 | Shift Line Insert | 0x1B4E |
| Shift-F6 | Shift Line Delete | 0x1B4F |
| Shift-F7 | Shift Character Insert | 0x1B71 |
| Shift-F8 | Shift Character Delete | 0x1B72 |

* To emulate the Televideo FUNCT key operation, press and release Shift-F2, then press and release the key you wish to use in conjunction with FUNCT. This will send the standard 3 byte FUNCT sequence: <SOH> <key pressed> <CR>.

Digital Equipment Corporation VT-52

ProComm supports the standard model VT-52 functions. Supported functions include: full/half duplex, keypad application mode, cursor control, erase functions, printer control functions and full display attributes.

ProComm responds to the Identify command (Esc Z) with Esc/Z (Standard VT-52 identification sequence). Keypad Application Mode functions are mapped to ProComm function keys as shown in the chart below.

| ProComm KEYS | DEC VT-52 FUNCTION | TRANSMITTED CODES |
|--------------------|--------------------------------------|-------------------|
| Tab | Horizontal Tab | 0x09 |
| Delete | Character Delete | 0x7F |
| Home | Home Cursor | 0x1B5B4B |
| Cursor Up | Cursor Up | 0x1B5B41 |
| Cursor Down | Cursor Down | 0x1B5B42 |
| Cursor Left | Cursor Left | 0x1B5B44 |
| Cursor Right | Cursor Right | 0x1B5B43 |
| Ctrl-PgDn | Clear screen | 0x1B5B481B5B324A |
| End | Erase end of line | 0x1B5B4B |
| Ctrl-Home | Insert Line | 0x1B5B4C |
| Ctrl-PgUp | Delete Line | 0x1B5B4D |
| Ctrl-J | Line Feed | 0x0A |
| F7 | Keypad Application mode 1 | 0x1B4F71 |
| F8 | Keypad Application mode 2 | 0x1B4F72 |
| Shift-F7 | Keypad Application mode 3 | 0x1B4F73 |
| F5 | Keypad Application mode 4 | 0x1B4F74 |
| Shift-F6 | Keypad Application mode 5 | 0x1B4F75 |
| Shift-F5 | Keypad Application mode 6 | 0x1B4F76 |
| F3 | Keypad Application mode 7 | 0x1B4F77 |
| F4 | Keypad Application mode 8 | 0x1B4F78 |
| Shift-F3 | Keypad Application mode 9 | 0x1B4F79 |
| F9 or F10..... | Keypad Application mode 0 | 0x1B4F70 |
| F1 | Program Function 1 (PF1) | 0x1B4F50 |
| F2 | Program Function 2 (PF2) | 0x1B4F51 |
| Shift-F1 | Program Function 3 (PF3) | 0x1B4F52 |
| Shift-F2 | Program Function 4 (PF4) | 0x1B4F53 |
| Shift-F4 | Keypad Application mode DASH | 0x1B4F6D |
| Shift-F5 | Keypad Application mode COMMA | 0x1B4F6C |
| Shift-F9 | Keypad Application mode PERIOD | 0x1B4F6E |
| Shift-F8 | Keypad Application mode ENTER | 0x1B4F4D |

Lear Siegler ADM 3/5

ProComm supports the standard ADM 3/5 series functions. Supported functions include: full/half duplex, full character set, erase functions, and cursor control.

| ProComm KEYS | ADM 3/5 FUNCTION | TRANSMITTED CODES |
|--------------------|-------------------------|-------------------|
| Tab | Horizontal Tab | 0x1B49 |
| Backtab | Reverse Tab | 0x1B51 |
| Home | Home Cursor | 0x1E |
| Cursor Up | Cursor Up | 0x1B |
| Cursor Down | Cursor Down | 0x0A |
| Cursor Left | Cursor Left | 0x18 |
| Cursor Right | Cursor Right | 0x1C |
| Ctrl-PgDn | Clear screen | 0x1A |
| End | Erase end of line | 0x1B54 |

Heath/Zenith 19

ProComm supports the standard Heath/Zenith 19 functions. Supported functions include: full/half duplex, full character set, program function keys, erase functions, cursor control, and display attributes.

| ProComm KEYS | H-19 FUNCTION | TRANSMITTED CODES |
|--------------------|----------------------------------|-------------------|
| Tab | Horizontal Tab | 0x09 |
| Home | Home Cursor | 0x1B595F5F |
| Cursor Up | Cursor Up | 0x1B41 |
| Cursor Down | Cursor Down | 0x1B42 |
| Cursor Left | Cursor Left | 0x1B44 |
| Cursor Right | Cursor Right | 0x1B43 |
| Ctrl-PgDn | Clear screen | 0x1B4C |
| End | Erase end of line | 0x1B45 |
| Ctrl-Home | Insert Line | 0x1B4C |
| Ctrl-PgUp | Delete Line | 0x1B4D |
| F1 | Program Function 1 (PF1) | 0x1B53 |
| F2 | Program Function 2 (PF2) | 0x1B54 |
| F3 | Program Function 3 (PF3) | 0x1B55 |
| F4 | Program Function 4 (PF4) | 0x1B56 |
| F5 | Program Function 5 (PF5) | 0x1B57 |
| F6 | Program Function 6 (PF6) | 0x1B50 |
| F7 | Program Function 7 (PF7) | 0x1B51 |
| F8 | Program Function 8 (PF8) | 0x1B52 |
| F9 | Program Function 9 (PF9) | 0x1B30 |
| F10 | Program Function 10 (PF10) | 0x1B31 |

ADDS Viewpoint

ProComm supports the standard ADDS Viewpoint functions. Supported functions include: full/half duplex, erase functions, inset/delete functions, cursor control, and display attributes.

| ProComm KEYS | ADDS FUNCTION | TRANSMITTED CODES |
|--------------------|-----------------------------|-------------------|
| Tab | Horizontal Tab | 0x09 |
| Home | Home Cursor | 0x1B595F5F |
| Cursor Up | Cursor Up | 0x01 |
| Cursor Down | Cursor Down | 0x0A |
| Cursor Left | Cursor Left | 0x15 |
| Cursor Right | Cursor Right | 0x06 |
| Ctrl-PgDn | Clear screen | 0x0C |
| End | Erase end of line | 0x1B4B |
| Ctrl-End | Erase end of screen | 0x1B6B |
| Ins | Insert Character | 0x1B46 |
| Ctrl-Home | Insert Line | 0x1B4D |
| Del | Delete Character | 0x1B45 |
| Ctrl-PgUp | Delete Line | 0x1B6C |
| F1 | Function 1 (F1) | 0x02310D |
| F2 | Function 2 (F2) | 0x02320D |
| F3 | Function 3 (F3) | 0x02330D |
| F4 | Function 4 (F4) | 0x02340D |
| F5 | Function 5 (F5) | 0x02350D |
| F6 | Function 6 (F6) | 0x02360D |
| F7 | Function 7 (F7) | 0x02370D |
| F8 | Function 8 (F8) | 0x02380D |
| Shift F1 | Shift Function 1 (F1) | 0x02190D |
| Shift F2 | Shift Function 2 (F2) | 0x02290D |
| Shift F3 | Shift Function 3 (F3) | 0x02390D |
| Shift F4 | Shift Function 4 (F4) | 0x02490D |
| Shift F5 | Shift Function 5 (F5) | 0x02590D |
| Shift F6 | Shift Function 6 (F6) | 0x02690D |
| Shift F7 | Shift Function 7 (F7) | 0x02790D |
| Shift F8 | Shift Function 8 (F8) | 0x02890D |

WYSE 100

ProComm supports the standard WYSE functions. Supported functions include: full/half duplex, erase functions, inset/delete functions, cursor control, and display attributes.

| ProComm KEYS | WYSE 100 FUNCTION | TRANSMITTED CODES |
|------------------|-----------------------------|-------------------|
| Tab | Horizontal Tab | 0x09 |
| Backtab | Reverse Tab | 0x1B49 |
| Insert | Insert Character | 0x1B51 |
| Ctrl-Home | Insert Line | 0x1B45 |
| Delete | Delete Character | 0x7F |
| Ctrl-PgUp | Delete Line | 0x1B52 |
| Home | Home Cursor | 0x1E |
| Cursor Up | Cursor Up | 0x1B |
| Cursor Down | Cursor Down | 0x0A |
| Cursor Left | Cursor Left | 0x1B |
| Cursor Right ... | Cursor Right | 0x1C |
| Ctrl-PgDn | Clear Screen | 0x1A |
| End | Line Erase | 0x1B54 |
| Ctrl-End | Page Erase | 0x1B59 |
| F1 | Function 1 (F1) | 0x01400D |
| F2 | Function 2 (F2) | 0x01410D |
| F3 | Function 3 (F3) | 0x01420D |
| F4 | Function 4 (F4) | 0x01430D |
| F5 | Function 5 (F5) | 0x01440D |
| F6 | Function 6 (F6) | 0x01450D |
| F7 | Function 7 (F7) | 0x01460D |
| F8 | Function 8 (F8) | 0x01470D |
| Shift F1 | Shift Function 1 (F1) | 0x01480D |
| Shift F2 | Shift Function 2 (F2) | 0x01490D |
| Shift F3 | Shift Function 3 (F3) | 0x014A0D |
| Shift F4 | Shift Function 4 (F4) | 0x014B0D |
| Shift F5 | Shift Function 5 (F5) | 0x014C0D |
| Shift F6 | Shift Function 6 (F6) | 0x014D0D |
| Shift F7 | Shift Function 7 (F7) | 0x014E0D |
| Shift F8 | Shift Function 8 (F8) | 0x014F0D |

ANSI-BBS

This is the recommended emulation for use with bulletin board systems that use ANSI graphics and color. The ANSI-BBS mode processes the ANSI codes given in the DOS technical reference manual, and behaves in the same manner as the DOS ANSI.SYS device driver. (ProComm does not use the ANSI.SYS driver itself). This emulation is similar to the VT-100 emulation, but the VT-100 emulation handles normal, bold, and reverse video in a different manner. The VT-100 keyboard mapping is used with this emulation.

APPENDIX B - COMMAND REFERENCE GUIDE

Major Functions

Dialing Directory Alt-D
Automatic Redial..... Alt-R
Keyboard Macros Alt-M
Line Settings Alt-P
Translate Table Alt-W
Editor Alt-A
Exit Alt-X
Host Mode Alt-Q
Chat Mode Alt-O
DOS Gateway Alt-F4
Command File Alt-F5
Redisplay Alt-F6

Utility Functions

Program Info Alt-I
Setup Screen Alt-S
Kermit Server Commands Alt-K
Change Directory Alt-B
Clear Screen Alt-C
Toggle Duplex Alt-E
Hang Up Phone Alt-H
Elapsed Time Alt-T
Print On/Off Alt-L
Set Colors Alt-Z
Auto Answer Alt-Y
Toggle CR - CR/LF Alt-F3
Break Key ALT-F7, Ctrl-Break

File Functions

Send Files PgUp
Receive Files PgDn
File Directory Alt-F
View a File Alt-V
Screen Dump Alt-G
Log Toggle Alt-F1
Log Hold Alt-F2

When I'm using a multi-tasking operating system and running ProComm in the background, window displays bleed through to the active partition.

Answer:

Go into the general SetUp screen and set ProComm to use the BIOS screen write method. It is much slower but will not cause the bleed through that you are experiencing.

Question:

When I try to use the DOS gateway it says 'Command processor not found'. What does this mean?

Answer:

In order to use the gateway, ProComm must load a secondary copy of the command processor (COMMAND.COM). It uses the environment variable COMSPEC to determine the name and location of the command processor. COMSPEC is set at system startup to the drive, directory and program you booted from. Problems can arise if you boot off a floppy, and then change the floppy in the boot disk drive. To use the gateway, and the Alt-F function, be sure that COMSPEC is set and that the command processor is where COMSPEC says it is.

Question:

I can't use the help screen because MultiLink grabs the ALT-F10 keystroke before ProComm can get it.

Answer:

MultiLink provides an 'escape' keystroke to avoid that problem. Simply press Alt-F9 before pressing Alt-F10 and MultiLink will let the Alt-F10 through to ProComm.

Question:

When I call online systems like CompuServe or the Source via Telenet, I get nothing but garbage characters on my screen.

Answer:

When using public networks such as Telenet or Tymnet, you must either call in at E/7/1 (even parity, 7 data and 1 stop bits), or strip the high bit off each incoming character using the translate table.

Question:

When I try to run Host Mode it always thinks that somebody is online; that is, it will automatically go to the "Name: " prompt and then keep cycling.

Answer:

You must set your modem so that it does not force carrier detect (CD) high, or true. This is usually controlled by a dip switch on your modem. For example, on a Hayes external mode, set dip switch 6 to the up position.

Question:

I'm trying to get two computers running ProComm to talk to each other using a direct connection. What do I do?

Answer:

You need to connect the two machines using a null modem cable. A null modem cable has the usual configuration for a serial port to modem connection, with the exception that pins 2 and 3 must be crossed. That is pin 2 on one end needs to go to pin 3 on the other, and vice versa. Then make sure that each side is using the same line parameters. You should be able to talk back and forth between the machines while in terminal, or chat modes (like Crosstalk's GO LOCAL). You may even run Host Mode on one machine; just remember to set the Connection Type option in the Host Mode Setup area to Direct.

Question:

ProComm doesn't work with my NeoClone modem that I bought in a garage sale in Taiwan. Why not?

Answer:

ProComm, by default, is set for use with Hayes compatible modems. If your modem is not truly Hayes compatible, you will have to change some of the SetUp Options, especially the Modem Initialization string and the Modem Dial command, to values correct for your modem. This may entail a little digging in your modem manual. User groups and local bulletin boards may be able to provide information on using your particular brand of modem.

APPENDIX D - PROCOMM TECHNICAL SPECIFICATIONS

Receive buffer size 2048 Bytes

Redisplay buffer size 10000 Bytes

Baud rates supported 300
..... 1200
..... 2400
..... 4800
..... 9600
..... 19200

File transfer protocols ASCII
..... XMODEM
..... WXMODEM
..... TELINK
..... YMODEM
..... MODEM7
..... KERMIT
..... COMUSERVE B

Serial ports supported COM1 0x3F8 IRQ4
..... COM2 0x2F8 IRQ3
..... COM3 0x3E8 IRQ4
..... COM4 0x2E8 IRQ3

Video buffer addresses supported COLOR 0xB8000
..... MONO 0xB0000

Interrupt vectors used 0x0B
..... 0x0C
..... 0x1B
..... 0x23

APPENDIX E - USER SUPPORTED SOFTWARE

User-supported software is a means for the computing community to receive quality software while directly supporting software authors. It is based on the ideas that:

The value and utility of software is best assessed by the user on his or her own system. Only after using a program can one really determine whether it serves personal applications, needs and tastes.

The creation of independent personal computer software can and should be supported by the computing community.

Copying of programs should be encouraged, rather than restricted. The ease with which software can be distributed outside traditional commercial channels reflects the strength, rather than the weakness, of electronic information.

Under the user supported concept, anyone may request a copy of a user-supported program by sending a blank, formatted disk to the program author together with an addressed, postage-paid return mailer. A copy of the program, along with documentation on disk, will be sent by return mail on the user's disk.

The program carries a notice suggesting registration for the program. You should register if you are going to use the program on a regular basis. Regardless of whether you register and use the program, you are encouraged to copy and distribute the program for the private, non-commercial, trial use of others.

User supported software is generally not public domain material; most programs of this nature carry a copyright notice. Rather, the author has licensed you to copy and use the program under certain conditions. Likewise, user supported software is not intended to be free software; it is an experiment in economics, not altruism. It is intended to provide quality software at a low price, while directly supporting the author, without the overhead of distributors, dealers and advertising that produces \$500 software packages.

User supported software is having a hard time. More and more packages are being taken out of this market, and offered as more traditional, and expensive, products. The reason for this is simple: lots of people are using the packages but very few are paying for them. And without the support of the users, there is absolutely no incentive for software authors to provide their programs in this fashion.

There are many good reasons to register. Besides supporting the author (that is, paying for the software you use), you generally get better support and receive mailed notification of updates and other products.

Some authors provide diskettes and documentation with registration; due to our low registration fee Datastorm Technologies, Inc. does not, although diskettes and manuals are available at a very low price (see Ordering Information in the front of this manual).

In conclusion, if you regularly use a user supported program (sometimes called Freeware or Shareware) and have not sent in a registration to the author, please do so now. Only through the financial support of users will this kind of inexpensive software continue to be available.

APPENDIX F - PRODUCT SUPPORT

In order to support our users, and to provide a means of distribution, we have implemented a bulletin board system you can use to communicate with us. If you have questions or comments you may call us up and leave a note. In your note, describe as completely as possible the problem you are having. Let us know your machine configuration, your ProComm configuration and version and any resident software you are using. Describe what steps you take before the problem occurs, and exactly what the program does when it occurs. If you do not provide us with a complete description of the problem there is little we can do to help. We'll do our best to keep you up and running, but if you are not a registered user we do not guarantee to provide support of any kind.

You'll always be able to find the latest version of ProComm on the BBS. The number for our board is (314) 449-9401. The board is operational 24 hours per day. Due to the incredibly high volume of calls that we are receiving, the BBS is often busy. You will probably need to put ProComm into auto redial and let it work for a while before you get in. We plan to install a second line for the benefit of registered users.

If you do go online to use, please remember a few things:

We have thousands of users to support, plus a product to maintain and a business to run. You will most likely not receive an immediate response. Please be patient. We usually take care of registered users in 2 or 3 days at the most.

We do not guarantee to provide support of any kind to non-registered users. We do, in fact, provide many hours of support to non-paying users, but we do this at our discretion. Non-supporting users who demand support from us are usually out of luck. Please be patient, and please be polite.

In addition to our board, several fine systems always carry the latest version of ProComm. They are:

Computer Aide BBS
Mike Johnson, Sysop
Tulsa, OK. (918) 493-2137
2400 baud, 24 hours a day

ATL/FIDO1
Ken Shackelford, Sysop
Woodstock, GA. (404) 928-1876
2400 baud, 24 hours a day

PConnecticut BBS
John O'Boyle, Sysop
West Hartford, CT. (203) 521-1991
2400 baud, 24 hours a day

TBC BBS
Dave Staehlin, Sysop
Albuquerque, NM, (505) 821-7379
2400 baud, 24 hours a day

Underdog BBS
Sal Manaro, Sysop
Seattle, WA. (206) 725-9233
2400 baud, 24 hours a day

Rowlett RBBS
Dan Kardell, Sysop
Rowlett, TX, (214) 475-4598
2400 baud, 24 hours a day

WELLSPRING BBS
Steve Clancy, Sysop
Irvine, CA, (714) 856-7996
1200 baud, 5pm-8am M-F,
24 hours weekends

IBM/PC SIG, The Source
Mike Todd, Sysop
The Source (800) 336-3330 (voice)
Mike Todd (213) 439-6104 (voice)

ProComm is also found on a host of other bulletin board systems, user groups and the like.

I N D E X

| | | | |
|-------------------|---------------|-----------------------------|--------------------|
| ! | 204, 312 | ASCII | 310, 602, 717 |
| +++ | 304, 313, 802 | ASCII Transfer SetUp | 314 |
| /B | 201 | ASSIGN Sx | 704 |
| /D | 201 | AT | 302, 304, 313, 802 |
| /F | 201, 701, 702 | ATDT | 303, 501 |
| /M | 201, 310 | Audit trail | 803 |
| /S | 201, 311 | Auto answer | 407, 802 |
| ^ | 204, 312 | Auto answer string | 313 |
| | 204, 312 | Auto baud detect | 313, 803 |
| ~ | 204, 311 | Auto redial | 304, 706 |
| ~~~ | 304, 313, 802 | Automatic redial | 401, 504 |
| Aborted downloads | 312 | BACKSPACE | 721 |
| ADDS Viewpoint | 909 | Batch transfers | 603 |
| Alarm sound | 311 | Baud rate | 203, 402 |
| Alarm time (secs) | 311 | BIOS | 310 |
| Alt-O | 307 | BIOS calls | 201 |
| Alt-A | 309, 403, 408 | Blank lines | 315 |
| Alt-B | 405 | Break | 407, 704 |
| Alt-C | 406 | Break Length (ms) | 307 |
| Alt-D | 303, 401 | BS key definition | 307 |
| Alt-E | 305, 406 | BS translation | 306 |
| Alt-F | 202, 408 | BYMODEM | 717 |
| Alt-F1 | 310, 409 | Carrier Detect | 101, 304 |
| Alt-F10 | 203, 205, 401 | CASE | 718 |
| Alt-F2 | 409 | CD | 304, 406, 711, 801 |
| Alt-F3 | 407 | CED | 804 |
| Alt-F4 | 404 | Character pacing | 315 |
| Alt-F5 | 405, 701 | Chat Mode | 404 |
| Alt-F6 | 404, 405 | CHDIR | 705 |
| Alt-F7 | 307, 408 | Checksum | 603 |
| Alt-G | 310, 408 | Christensen, Ward | 603 |
| Alt-H | 304, 406, 710 | CIS B | 308 |
| Alt-I | 405 | CISB | 717 |
| Alt-K | 405, 604 | CLEAR | 705, 714 |
| Alt-L | 406, 409 | Clear screen | 406 |
| Alt-M | 401 | CLOSE | 714 |
| Alt-O | 404 | Command File Syntax | 702 |
| Alt-P | 402, 404 | Command files | 405, 701 |
| Alt-Q | 404, 803 | Error messages | 724 |
| Alt-R | 401, 504 | Command line switches | 201 |
| Alt-S | 301, 405, 406 | Command Reference Guide | 1001 |
| Alt-V | 408 | COMMAND.COM | 403, 404, 408, 706 |
| Alt-W | 310, 403 | Comment | 703 |
| Alt-X | 205, 404 | Commercial use | iii |
| Alt-Y | 407 | Composite monitor | 201 |
| Alt-Z | 406, 407 | CompuServe | 311, 403, 603 |
| ANSI.SYS | 911 | CompuServe 'B' protocol | 308 |
| ANSI-BBS | 305, 911 | CompuServe B File Transfers | 604 |

| | | | |
|--------------------------------------|--------------------|-------------------------------------|--------------------|
| COMSPEC | 404, 408, 706 | EMULATE | 706 |
| COMSWAP.ARC | 1101 | End key | 405, 501, 701 |
| CONFIG.SYS | 101, 801 | ENDCASE | 718 |
| Connect string | 303 | ENDSWITCH | 718 |
| CONNECTED | 711 | END | 307, 604 |
| Connection type | 314 | Enquiry | 307 |
| Copying ProComm | iii | Environment Variable | 102 |
| Copyright | iii | Error messages | 724 |
| CR | 407 | ESC translation character | 204 |
| CR translation | 306, 315, 316 | EXECUTE | 707 |
| CR translation character | 204 | Exit | 404, 707 |
| CR/LF | 306 | Exiting ProComm | 205 |
| CRC | 603 | Expand blank lines | 315 |
| CTRL character translation | 204 | Exploding windows | 311 |
| CTRL-BREAK | 307, 407 | FAILURE | 711 |
| Ctrl-E | 307 | FDX | 406 |
| Ctrl-G | 311 | FIDO | 603 |
| Ctrl-J | 303 | File logging | 409, 714 |
| Ctrl-M | 303 | File Transfer Protocols | 602 |
| Ctrl-Q | 404 | FILES= | 101, 802 |
| Ctrl-S | 404 | FIND | 707 |
| CTTY COMx | 804 | FINISH | 713 |
| CWHEN | 720 | Flow control | 305 |
| Data bits | 203, 402 | FOUND | 711, 712 |
| Data Terminal Ready | 304 | General SetUp | 308 |
| Deadlock | 306 | GET | 708 |
| DEC VT100 | 305 | GETFILE | 708, 713 |
| Default d/l path | 309 | GOSUB | 709 |
| Default directory | 405 | GOTO | 710 |
| Default log file | 310 | HO | 304 |
| DEL | 307 | Handshaking | 305 |
| DIAL | 705 | Hang-up string | 304 |
| Dialing | 504 | Hangup | 404, 406, 710 |
| Dialing command | 303 | Hardware | 101 |
| Dialing command suffix | 303 | HDX | 406 |
| Dialing directory | 401, 501 | Heath/Zenith 19 | 908 |
| Adding entries | 502 | Help Screen | 202, 203, 205 |
| Deleting entries | 503 | Home key | 405, 408, 501, 701 |
| Making a call | 504 | HOST | 710 |
| Manual dialing | 504 | Host Access Password | 803 |
| Revising | 502 | Host ID string | 313, 802 |
| DIR | 408 | Host menu | 803 |
| DOS | 706 | Host Mode | 102, 404, 801 |
| DOS Gateway | 309, 404, 408 | Host mode password | 313 |
| DOS shell password | 313, 803 | Host Mode SetUp | 312 |
| Dosedit | 804 | IBM 3101 | 904 |
| Download | 408, 708 | IF | 711 |
| Download Path | 602 | IF CONNECTED | 710 |
| Downloading Files | 601 | IMG file | 408 |
| DTR | 304, 406, 801 | ISFILE | 713 |
| Duplex | 203, 305, 402, 406 | KERMIT | 717 |
| EO | 302 | Kermit File Transfer | 603 |
| Echo | 305 | Kermit handshake char | 308 |
| Echo locally | 314 | Kermit server commands | 405 |
| Editor | 309, 403 | Kermit SetUp | 308 |
| Elapsed time | 406 | KERMESERVE | 713 |
| ELSE | 711 | KEY files | 402 |

| | | | |
|---------------------------------------|---------------|--------------------------------------|---------------|
| KEY HIT | 314, 803 | PgDn | 405, 408, 501 |
| Keyboard macros | 401 | PgUp | 405, 408, 501 |
| KFLUSH | 714 | Print logging | 715 |
| Labels | 703 | Print on/off | 406 |
| LD code identifier | 502 | PRINTER | 715 |
| Lear Siegler ADM 3/5 | 907 | Printer logging | 406, 409 |
| LF | 407 | PRN | 409, 715 |
| LF translation | 316 | ProComm Files | 102 |
| LF translation (uploads) | 316 | PROCOMM.DIR | 102 |
| LICENSE | iii | PROCOMM.HST | 102 |
| Licensing agreements | vii | PROCOMM.KEY | 102 |
| Line pacing | 315 | PROCOMM.MSG | 102 |
| Line Settings | 203, 402 | PROCOMM.PRM | 102 |
| Line wrap | 307 | PROCOMM.XLT | 102 |
| LINKED | 711 | ProComm Setup | 802 |
| LOCATE | 714 | PROCOMM.DIR | 102 |
| LOG | 714 | PROCOMM.HST | 102, 803 |
| Log Hold | 408, 409 | PROCOMM.IMG | 408 |
| LOGOUT | 713 | PROCOMM.KEY | 102, 402 |
| Long Distance Codes | 503 | PROCOMM.MSG | 102, 803 |
| Adding or revising | 503 | PROCOMM.PRM | 102 |
| MACRO | 714 | PROCOMM.XLT | 102 |
| Manual Dialing | 504 | PROCOMM= | 102 |
| MESSAGE | 715 | PROFILE.CMD | 701 |
| MGET | 708 | Program information | 405 |
| MLOAD | 402, 714, 715 | Program information screen | 202 |
| MODEM | 717 | PRT OFF | 406 |
| Modem auto-answer string | 802 | PRT ON | 406 |
| Modem dialing command | 503 | Public domain | iii |
| Modem initialization string | 302, 803 | QUIT | 715 |
| MODEM MSG | 314, 803 | RAM requirements | 101 |
| Modem pause character | 205 | Receive buffer | 703 |
| Modem pause delay | 505 | Receive Files | 408 |
| Modem Setup | 302, 801 | Redial Pause Delay | 304 |
| MODEM7 File Transfers | 603 | Redial Timeout Delay | 304, 505 |
| Multi-tasking | 1102 | Redisplay | 404, 405 |
| Multi-tasking operating | 201 | Registered users | iii |
| systems | 201 | Registration | v |
| MultiLink | 1102 | RESUME | 714 |
| Nesting | 703 | RETURN | 709 |
| No Connect strings | 304 | RFLUSH | 716 |
| Non-registered users | iii | RGET | 716 |
| NOT | 712 | RUN | 716 |
| OPEN | 714 | RXMODEM | 717 |
| Operating System Setup | 801 | SO | 302 |
| Order | vi | SO= | 313, 802 |
| ORDERING INFORMATION | v | S11 | 302 |
| Pace character | 315 | S7 | 302 |
| Pacing | 315 | Screen buffer | 310 |
| Parity | 203, 402 | Screen dump | 408, 718 |
| Partial files | 312 | Screen dump file | 310 |
| Password | 313 | Screen write method | 310 |
| PAUSE | 715 | Scroll | 307 |
| Pause translation character | 204 | Searching for an Entry | 502 |
| PC-HOST BBS | 604 | Security | 313 |
| PCjr | 303, 1101 | Send Files | 408 |
| PeopleLink | 605 | SENDFILE | 713, 717 |

SET 717
 SET ALARM ON/OFF 721
 SET ASCII 721
 Set ASCII Commands 722
 SET ATIME 721
 SET BAUDRATE 721
 Set colors 406
 Set Commands 721
 SET CR_IN CR/CR_LF 721
 SET CR_OUT CR/CR_LF 721
 SET DATABITS 7/8 722
 SET DLDIR 722
 SET ENQ 722
 SET FLOWCTRL ON/OFF 722
 SET HOSTPSWD 722
 Set Kermit Commands 723
 SET PARITY 722
 SET PORT 722
 SET RDELAY 722
 SET SCROLL ON/OFF 722
 SET SHELLPSWD 722
 SET SOUND ON/OFF 722
 SET STOPBITS 1/2 722
 SET SWRITE BIOS/DIRECT 722
 SET TRANSLATE ON/OFF 722
 SET WRAP ON/OFF 722
 SetUp 301
 Setup menu 403
 Setup screen 405
 Sliding windows 604
 SNAPSHOT 718
 Sound effects 311
 Source, The 403, 604
 Split screen 404
 Sprint 503
 Stack overflow 709, 724
 Stack underflow 709, 724
 Status line 202, 401
 Stop bits 203, 402
 String Translation 204
 String variables 703
 Strip 310, 316
 Strip characters 403
 SUCCESS 711
 SUSPEnd 714
 SWITCH 718
 Sysops iii
 TCOMM BBS 604
 Telenet 403
 Televideo 900 Series 905
 TELINK 717
 Telink File Transfers 603
 Terminal Emulation 305
 Terminal Emulations 203
 Terminal Mode 202, 401
 Terminal SetUp 305
 Terminal types 707

Tilde 311
 Translate CR character 312
 Translate CTRL character 312
 Translate ESC character 312
 Translate pause character 311
 Translate table 310, 403
 TRANSMIT 720
 Truncated lines 307
 Tymnet 403
 Upload 408, 717
 Uploading Files 601
 V1 302
 VAX/VMS EDT Editor 903
 View a File 408
 VT-100 902
 VT-102 902
 VT-52 906
 WAITFOR 711, 712, 720
 WARRANTY iv
 WHEN 720
 Windows 311
 WXMODEM File Transfers 604
 WYSE 100 910
 X1 302
 XLT file 403
 XMODEM 310, 311, 717
 XMODEM File Transfer 603
 XMODEM timeouts 311
 XON/XOFF 306, 404, 602
 YMODEM 310, 717
 YMODEM Batch 603
 YMODEM File Transfers 603

TRUNKING MEMORY BACKUP

AND

THE IBM-PC

PRESENTED BY

MARVIN GOODMAN-FTR

MOTOROLA C & E, HOUSTON



THE PC-TALK III PROGRAM WILL BE PLACED IN DRIVE A: AND THE DATA DISK WILL BE PLACED IN DRIVE B: AN EXAMPLE DIRECTORY IS INCLUDED ON HOW TO SET UP A PHONE DIRECTORY FILE WHICH WILL UTILIZE DIFFERENT METHODS FOR SAVING, VERIFING, AND LOADING SYSTEM DATA.

EACH TIME DATA IS SAVED FROM THE SYSTEM, WHETHER IT IS "SUBSCRIBER" OR INTERCONNECT, THE EXISTING FILES ON DRIVE B: WILL NEED TO BE DELETED. IF THIS IS NOT DONE, THEN NEW DATA WILL BE ADDED TO THE END OF THE DATA ALREADY ON DISK.

III. OPERATION: "SUBSCRIBER" RAM BACKUP--CSC09 BOARD

THE COMMAND LIST, ENTERED FROM THE IBM-PC, IS AS FOLLOWS:

SAVE THIS COMMAND ALLOWS THE SYSTEM MANAGER TO SAVE ALL THE SUBSCRIBER RECORDS AND RELATED INFORMATION. THE SAVE, LOAD, AND VERIFY ROUTINES ARE DESIGNED TO INTERACT WITH THE IBM-PC, WHICH WILL RESPOND TO DEVICE CONTROL CODES, PRIMARILY CONTROL-Q.

LOAD THIS COMMAND IS USED TO LOAD THE "SUBSCRIBER" RECORDS FROM THE IBM-PC TO SYSTEM MEMORY. WHEN "LOAD" IS TYPED, THE OPERATOR IS REMINDED TO MAKE SURE THE COMPUTER IS READY. WHEN IT IS, A CARRIAGE RETURN TELLS SYSTEM TO START READING THE DATA. THE SYSTEM WILL SEND A "CTRL-Q" AND THE IBM WILL READ OUT THE FIRST S-RECORD, STOPPING TO WAIT FOR ANOTHER "CTRL-Q" FROM THE SYSTEM TO THE IBM TO READ OUT ANOTHER LINE. AS THE DATA IS READ IN, THE SYSTEM WILL ECHO BACK SO IT WILL APPEAR ON THE SCREEN. THIS CONTINUES UNTIL THE S-RECORD INDICATING THE END-OF-FILE IS READ. IF A CHECKSUM ERROR IS DETECTED IN AN S-RECORD, OR THE OPERATOR TYPES A CTRL-X, THE LOAD WILL BE ABORTED AND THE SYSTEM WILL JUMP TO THE STATUS ROUTINE. A SUCCESSFUL LOAD RETURNS TO THE "SUBS>" PROMPT. "SUBSCRIBER" FUNCTION IS SET TO INACTIVE WHEN A LOAD COMMAND IS USED.

VERIFY THE VERIFY COMMAND IS USED TO COMPARE THE "SUBSCRIBER" RECORDS IN MEMORY WITH WHAT IS SAVED ON DISK. THE OPERATION IS BASICALLY THE SAME AS FOR THE LOAD COMMAND, EXCEPT THE DATA IN THE S-RECORD IS COMPARED WITH MEMORY INSTEAD OF STORED IN MEMORY. IF THE DATA IS NOT THE SAME A



CHECKSUM ERROR IS DETECTED IN THE S-RECORD, OR A CTRL-X IS TYPED, THE OPERATION WILL ABORT AND THE "SUBS>" PROMPT WILL BE PRINTED.

QUIT THE QUIT COMMAND IS USED TO EXIT "SUBSCRIBER" AND RETURN TO THE GUARDIAN PROMPT "GRD>".

COMMAND PROCEDURE

UPON SYSTEM ACKNOWLEDGE THE SYSTEM PASSWORD WILL NEED TO BE ENTERED, AND THE "GRD>" PROMPT WILL APPEAR. "SUBS" IS THEN TO BE TYPED; WHEREUPON, A SECOND PASSWORD IS TO BE ENTERED. THE DEFAULT PASSWORD, AFTER INITIAL INSTALLATION OF S.A.C SOFTWARE IS THE SYSTEM NUMBER. THIS CAN BE CHANGED AND SHOULD BE CHANGED TO A PASSWORD OF THE SYSTEM MANAGER'S CHOICE. THIS WILL PREVENT UN-AUTHORIZED PERSONS FROM CORRUPTING THE "SUBSCRIBER" DATA.

A WORD OF CAUTION

DON'T TOUCH THE KEYBOARD ONCE THE DATA STREAM HAS BEGUN DURING A VERIFY OR LOAD SEQUENCE. THIS ACTION IS PRESENTED TO THE SYSTEM AS CORRUPT DATA AND A "BACKUP" ERROR WILL OCCUR.

TO SAVE, VERIFY, AND LOAD MEMORY

A. SAVE - TO SAVE DATA TO DISK

NOTE FOLLOW DIRECTIONS IN THE PC-TALK III DOCUMENTATION AS TO STRIPPING ASCII 19 AS THE DATA IS DOWNLOADED. THE LOAD AND VERIFY PHASES OF THIS PROCEDURE WILL NOT WORK WITHOUT FOLLOWING THOSE DIRECTIONS

STEPS: - GRD>SUBS

ENTER PASSWORD

OPTION LISTINGS WILL APPEAR

SUBS>SAVE

HIT RETURN WHEN BACKUP UNIT IS READY

ALT-R: B:SAC.RCD (FILE SPEC.) <CR>

AT THIS TIME A STREAM OF DATA WILL BEGIN BEING PRINTED TO THE SCREEN AS WELL AS IT BEING LOADED TO THE DISK IN DRIVE B. ONCE THE SAVE IS COMPLETE, THE SYSTEM WILL SEND A "SAVE DONE" MESSAGE TO THE SCREEN. HIT ALT-R TO CLOSE THE SAC.RCD FILE. THE NEXT STEP IS TO VERIFY THE DATA.



B. VERIFY - TO VERIFY THE DATA "BACKED UP"

SUBS>VERF

HIT RETURN WHEN BACKUP UNIT IS READY

ALT-T: B:SAC.RCD <CR> =P< <CR> RETRIEVED

THE DATA FROM DRIVE B: WILL BEGIN TO BE SENT TO THE SYSTEM FOR VERIFICATION. IT WILL BE ECHOED BACK ON THE SCREEN. IF DATA IS NOT THE SAME A CHECKSUM ERROR WILL BE DETECTED, THE OPERATION WILL ABORT AND THE "SUBS>" PROMPT WILL PRINTED ON THE SCREEN. IF THIS HAPPENS, DON'T PANIC, SIMPLY TRY TO VERIFY THE DATA AGAIN. IF A CHECKSUM ERROR IS DETECTED AGAIN, THEN THE DATA IS CORRUPT AND A "SAVE" WILL NEED TO BE COMPLETED ONCE MORE. NOISY PHONE LINES MAY CAUSE THE DATA TO BE CORRUPTED WHILE IT IS BEING RECEIVED OR TRANSMITTED. ONCE THE VERIFY IS COMPLETE THE "SUBS>" PROMPT WILL RETURN.

C. LOAD - LOADING A DATA DISK TO THE SYSTEM

SUBS>LOAD

HIT RETURN WHEN BACKUP UNIT IS READY

ALT-T: B:SAC.RCD <CR> =P< <CR> RETRIEVED

THE DATA FROM DRIVE B: WILL BEGIN TO BE SENT TO THE SYSTEM. IT WILL BE ECHOED BACK TO THE SCREEN. IF AN ERROR OCCURS IT CAN BE READILY SEEN ON THE SCREEN. THE LOAD WILL BE ABORTED AND THE SYSTEM WILL JUMP TO THE "STATUS" ROUTINE. REMEMBER, DATA IS FACED FROM THE IBM-PC UPON THE RECEIPT OF A CTRL-Q, WHICH = <.

IV. OPERATION: MCB RAM BACKUP--INTERCONNECT MEMORY

HERE IS A LIST OF COMMANDS AS ENTERED FROM THE IBM PC:

- P x-y "PUNCH" MEMORY CONTENTS SPECIFIED ONTO DISK.
x=FROM ADDRESS, y=TO ADDRESS.
- L LOAD DISK CONTENTS INTO MEMORY
- V VERIFY THAT DISK EQUALS THE MEMORY
- Q QUIT THE GUARDIAN DISK ROUTINES AND RE-ENABLE
CALL PROCESSING
- WE WRITE ENABLE DYNAMIC RAM SO A DISK CAN BE LOADED
INTO MEMORY



IL ENABLE INTERNAL LOW PAGE OF DYNAMIC RAM
IH ENABLE INTERNAL HIGH PAGE OF DYNAMIC RAM
TS SET THE TAPE PORT TO BE THE SAME AS THE SYSTEM
 MANAGER PORT.
TD SET THE TAPE PORT BACK TO THE AUX. PORT J4

COMMAND PROCEDURE

WHEN CALLING THE TRUNKING SYSTEM REMOTELY AND AFTER ENTERING THE "BILL" SECTION, A "?" WILL APPEAR. TYPE "GRDD" AND A "!" WILL APPEAR MOMENTARILY. THE MODEM AT THE CENTRAL SITE MAY HANG UP IN THE PROCESS OF SHIFTING FROM "BILL" TO THE GUARDIAN TAPE ROUTINE. THIS IS NORMAL SINCE THE MCB GOES THROUGH A RESET IN THE PROCESS OF TRANSFERRING CONTROL. SIMPLY CALL IT BACK. ONCE THE GUARDIAN PROMPT (!) STATE HAS BEEN REACHED, ANY OF THE COMMANDS LISTED ABOVE CAN BE ENTERED.

TO STORE, VERIFY, AND LOAD MEMORY

A. PUNCH - TO MAKE A DISK.

STEPS: GRD>BILL

?GRDD (IT MAY BE NECESSARY TO CALL THE SYSTEM
 BACK AT THIS POINT!!)

!TS

!IL

!P 0-7FFF - SELECTS PAGE 0 OF DYNAMIC RAM
 ALT-R: IL.RCD #1 <CR> <CR>
 WHEN (!) REAPPEARS THEN ALT-R TO END REC.

!IH - SELECTS INTERNAL HIGH PAGE OF DYNAMIC RAM

!P 0-7FFF - SELECTS PAGE 1 OF DYNAMIC RAM
 ALT-R: IH.RCD #2 <CR> <CR>
 WHEN (!) REAPPEARS THEN ALT-R TO END REC.

!P 9000-9FFF - SELECTS STATIC RAM
 ALT-R: STATIC.RCD #3 <CR> <CR>
 WHEN (!) REAPPEARS THEN ALT-R TO END REC.

!Q - LEAVING GUARDIAN AND CALL PROCESSING WILL BEGIN



B. VERIFY-TO VERIFY A DISK

STEPS: VERIFICATION IS TO BE DONE IMMEDIATELY AFTER DISK STORAGE. DO NOT TYPE "Q" TO LEAVE GUARDIAN, BUT CONTINUE AS FOLLOWS:

!TS

!IL

!V <CR>

◀ ALT-T: IL.RCD #1 <CR>

WHEN THE FILE IS COMPLETE THE TX WILL END FOR YOU AUTOMATICALLY.

!IH

!V <CR>

◀ ALT-T: IH.RCD #2 <CR>

!V <CR>

◀ ALT-T: STATIC.RCD #3 <CR>

!Q

C. LOADING A DISK BACK INTO THE MCB

STEPS: GRD>BILL

?GRDD

!TS

!WE

!IL

!L <CR>

◀ ALT-T: IL.RCD #1 <CR>

WHEN THE FILE IS COMPLETE THE TX WILL END AUTOMATICALLY.

!IH

!L <CR>

◀ ALT-T: IH.RCD #2 <CR>

!L <CR>

◀ ALT-T: STATIC.RCD #3 <CR>

!Q

LEAVING GUARDIAN

IN THE
EIGHT MI



NOTE: WHERE <CR> IS TYPED, PLEASE BE CERTAIN TO DO A CARRIAGE RETURN AS MANY TIMES AS NOTED. UN-MARKED ENTRIES ASSUME A CARRIAGE RETURN. AN UNDERSTANDING OF "WHY" WILL BE KNOWN IF THE CARRIAGE RETURNS ARE ENTERED IMPROPERLY. THE IL.RCD, IH.RCD, AND STATIC.RCD ARE INDICATORS OF FILE NAMES. "RCD" INDICATES "RECORD". THE NUMBERS #1, #2, AND #3 INDICATE THE FILES NECESSARY AND ARE NOT TO BE TYPED IN.

ERROR MESSAGES

IF WHILE VERIFYING THE DATA ON DISK WITH THE DATA IN RAM AND A MISMATCH OF DATA OCCURS, IT WILL NOT BE KNOWN UNTIL THE VERIFY IS COMPLETE. ONCE COMPLETE, AND THERE WAS A MISMATCH OF DATA, THE ERROR MESSAGE "MEMORY STORAGE ERROR" OR "CHECKSUM ERROR" WILL APPEAR ON THE SCREEN.

DON'T PANIC - TRY THE VERIFY ROUTINE AGAIN. IT MAY HAVE BEEN A NOISY PHONE LINE WHICH DROPPED A DATA BIT. IF THE ERROR MESSAGE COMES BACK AGAIN THEN RE-PUNCH THE DISK. BUT, REMEMBER, THIS MEANS MORE TIME FOR DISABLED INTERCONNECT.

IF WHILE LOADING DATA TO THE MCB THERE IS A PROBLEM, IT WILL NOT BE KNOWN UNTIL THE LOAD IS COMPLETE. IF THE LOAD FAILED, THEN THE MESSAGE "TAPE FAILURE" OR "MEMORY STORAGE ERROR" WILL APPEAR ON THE SCREEN.

DON'T PANIC - TRY THE LOAD SEQUENCE AGAIN. IF THE ERROR COMES BACK THEN THE DATA BASE CONTENTS COULD BE MODIFIED AT THIS POINT REQUIRING THE DATA BE TYPED IN TO GUARDIAN IN THE CONVENTIONAL MANNER.

V. SUMMARY

THE DESCRIBED OPERATION OF "BACKING UP" SYSTEM MEMORY IS AN ON-GOING PROCESS. EACH TIME PROGRAMMING IS ADDED OR MODIFIED IN THE SYSTEM, IT WILL NEED TO BE "BACKED UP". IF THE SYSTEM MEMORY BECOMES LOST, THEN UP-TO-DATE DATA CAN BE LOADED TO THE SYSTEM.

THERE IS A TREMENDOUS AMOUNT OF DOCUMENTATION PRESENTED HERE AND UNTIL IT IS IMPLEMENTED SEVERAL TIMES AND UNDERSTOOD, THERE MAY BE PROBLEMS. CALL ME! I'LL HELP WORK OUT THE BUGS AND TRY TO PROVIDE BETTER UNDERSTANDING ON THE METHOD OF OPERATION. I AM LOCATED AT MOTOROLA'S AREA OFFICE IN HOUSTON. MY PHONE NUMBER IS 713-537-3635.

PC-TALK III IS A PRODUCT OF "FREWARE SOFTWARE" AND IS FREE TO THE USER. A CONTRIBUTION OF \$35.00 IS REQUESTED BUT NOT MANDATORY. "A SMALL PRICE TO PAY FOR SUCH A POWERFUL PROGRAM". THE INFORMATION FOR THIS CONTRIBUTION IS PRESENTED IN THE PC-TALK DOCUMENTATION SECTION. GOOD LUCK!



COMMUNICATIONS PARAMETERS PC-TALK III

WITHIN THE "SUBSCRIBER" SECTION OF THE TRUNKING SOFTWARE, THERE IS A CERTAIN CONTROL CODE (CONTROL-S) SENT FROM THE SYSTEM WHICH IS USED TO CONTROL THE OPERATION OF A DATA RECORDER DURING LOAD AND VERIFY FUNCTIONS. THIS CODE IS A "STOP READ" CODE FOR THE RECORDER TO STOP READING OUT EACH LINE AND WAIT FOR A "START READ" CODE FROM THE SYSTEM. WHEN USING THE IBM-PC AS THE BACKUP TOOL, THIS CODE IS WRITTEN ON THE DATA DISK. THIS IS UNSATISFACTORY BECAUSE DURING UP LOAD THIS CODE IS SENT BACK TO THE SYSTEM. THE SYSTEM DOES NOT KNOW WHAT TO DO WITH THIS CODE AND AFTER A CERTAIN AMOUNT OF TIME A "BACKUP" ERROR WILL OCCUR. IT IS THEN NECESSARY TO "STRIP" THIS ASCII CODE FROM THE INCOMING DATA, SO IT WILL NOT BE WRITTEN TO DISK. THE FOLLOWING PAGES HAVE THE NECESSARY EVENTS ON PC-TALK TO IMPLEMENT THE "STRIPPING" OPTION. THE CODE TO STRIP WILL BE AN ASCII 019. FOLLOW THE DOCUMENTATION AND THE EVENTUAL LOAD AND VERIFY SEQUENCE TO THE SYSTEM WILL WORK FINE. A COPY OF "FREWARE'S" PC-TALK III IS INCLUDED FOR IN DEPTH UNDERSTANDING OF THIS COMMUNICATIONS PROGRAM.

IT IS NECESSARY TO CREATE TWO TYPES OF PHONE DIRECTORY FILES FOR OPTIMUM OPERATION.

FILE #1----THIS DIRECTORY FILE IS TO BE CREATED FOR THE "SUBSCRIBER" SECTION OF THE SOFTWARE. IT INCLUDES THE "PACING" AND "STRIPPING" OPTIONS REQUIRED.

WHEN ADDING THE "PACING" OPTION, IT NECESSARY TO HIT THE "CONTROL-Q" KEYS WHEN ASKED P=. HIT RETURN AND THE "PACING" CHARACTER WILL APPEAR AS SHOWN.

FILE #2---THIS FILE IS CREATED FOR USE WITH THE "INTERCONNECT" DATA COMING FROM AND GOING TO THE SYSTEM. NO SPECIAL "PACING" OR "STRIPPING" OPTIONS ARE REQUIRED.

THE DEFAULT SHEET LISTS THE PROGRAM DEFAULT OPTIONS AS THEY ARE TO BE PROGRAMMED. PRIMARILY, THEY WILL NOT COME INTO PLAY AS LONG AS THE DIRECTORY OPTIONS ARE PROGRAMMED.

How do I low-level format an MFM or RLL drive?

A low-level format actually does a physical format; it lays down tracks and sectors. It is the first format for a hard disk, and is used later to extend the life of an old drive to refresh the media. It wipes all information, just like reformatting a floppy disk.

Doing a regular format on a hard drive merely re-organizes the File Allocation Table (FAT), and doesn't really erase the disk or lay down tracks and sectors.

Spinrite, mentioned below, actually reads information off the disk, and re-writes it as it finishes formatting sectors on the disk, which is why it is not destructive like an ordinary low-level format. The instructions below work for small-disk Western Digital and Adaptec MFM and RLL controllers. Newer controllers usually use bundled software for low-level formatting.

WARNING: DO NOT LOW-LEVEL FORMAT IDE DRIVES!

It is best to do this from a bootable floppy. The floppy should have DEBUG.COM, FDISK.COM and FORMAT.COM on it.

The old 10MB IBM XT drives have to be low-level formatted using a routine on the IBM Advanced Diagnostics diskette. Zenith machines use the PREP command in Zenith DOS. Other older machines require use of the debug utility.

At the A:\> prompt, type:

```
debug
```

You will get a "-" prompt. At this point, type:

```
G=C800:5 -or-    for Western Digital controllers  
G=C800:800
```

```
G=C800:CCC      for Adaptec controllers.
```

```
G=C800:5        for DTC (Data Technonolgy) controllers
```

```
G=C800:6        for OMTI controllers
```

Spinrite is a good alternative to ordinary low-level formatting, because it doesn't destroy the contents of a disk. However, it does not work with many kinds of drives, especially those that use sector translation.

How do I low-level format an MFM or RLL drive?

A low-level format actually does a physical format; it lays down tracks and sectors. It is the first format for a hard disk, and is used later to extend the life of an old drive to refresh the media. It wipes all information, just like reformatting a floppy disk.

Doing a regular format on a hard drive merely re-organizes the File Allocation Table (FAT), and doesn't really erase the disk or lay down tracks and sectors.

Spinrite, mentioned below, actually reads information off the disk, and re-writes it as it finishes formatting sectors on the disk, which is why it is not destructive like an ordinary low-level format. The instructions below work for small-disk Western Digital and Adaptec MFM and RLL controllers. Newer controllers usually use bundled software for low-level formatting.

WARNING: DO NOT LOW-LEVEL FORMAT IDE DRIVES!

It is best to do this from a bootable floppy. The floppy should have DEBUG.COM, FDISK.COM and FORMAT.COM on it.

The old 10MB IBM XT drives have to be low-level formatted using a routine on the IBM Advanced Diagnostics diskette. Zenith machines use the PREP command in Zenith DOS. Other older machines require use of the debug utility.

At the A:\> prompt, type:

```
debug
```

You will get a "-" prompt. At this point, type:

```
G=C800:5 -or-    for Western Digital controllers  
G=C800:800
```

```
G=C800:CCC      for Adaptec controllers.
```

```
G=C800:5        for DTC (Data Technonolgy) controllers
```

```
G=C800:6        for OMTI controllers
```

Spinrite is a good alternative to ordinary low-level formatting, because it doesn't destroy the contents of a disk. However, it does not work with many kinds of drives, especially those that use sector translation.

DESCRIPTION

The model 70-1308A assembly is used to interface an IBM®PC/XT/AT or compatible computer to a Syn-Tech II, Syn-Tech XTR, or 8-Channel mobile transceiver, or to a Syn-Tech XTR Handheld for programming or test purposes.

If you are programming a mobile transceiver, the 25-pin connector plugs into a serial port of the computer while the 10-pin connector plugs directly into the transceiver, and the 70-1308A is powered by the transceiver.

If you are programming a Syn-Tech XTR Handheld, you will also need to connect the 70-1053A interface cable, the 70-1056A universal test box, or the 70-1057A programming/RF test cable to the radio. The 25-pin connector of the 70-1308A plugs into a serial port of the computer while the 10-pin connector plugs directly into the 70-1053A, 70-1056A, or 70-1057A.

Associated Manuals:

| | |
|-----------|---|
| 70-999375 | Computer-Based Programmer for Syn-Tech II Mobile Transceiver User's Manual |
| 70-999598 | Computer-Based Programmer for Syn-Tech XTR Mobile Transceiver User's Manual |
| 70-999993 | Computer-Based Programmer for 8-Channel Mobile Transceiver User's Manual |
| 70-999970 | Computer-Based Programmer for Syn-Tech XTR Handhelds User's Manual |

*IBM is the registered trade mark of the International Business Machine Corporation.

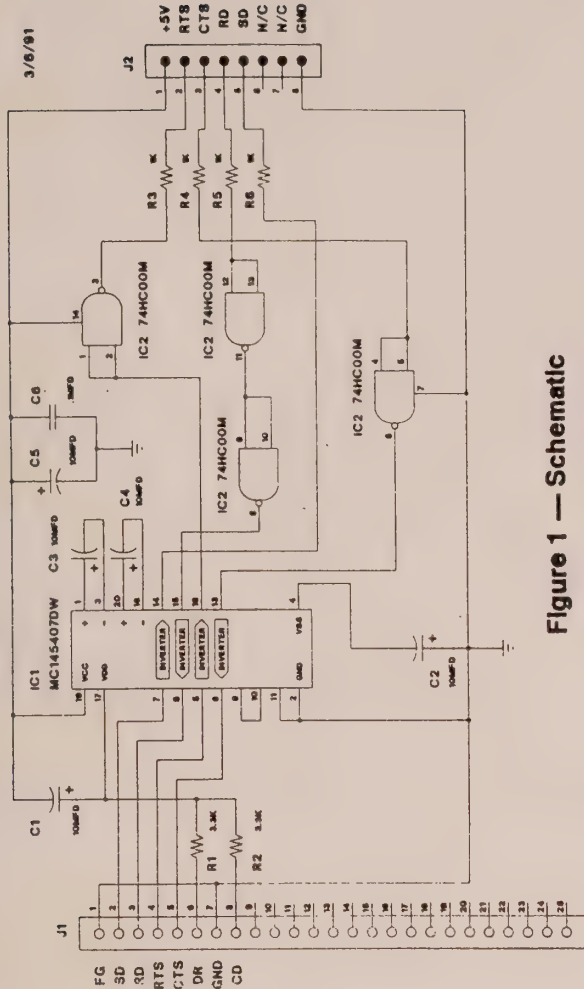


Figure 1 — Schematic

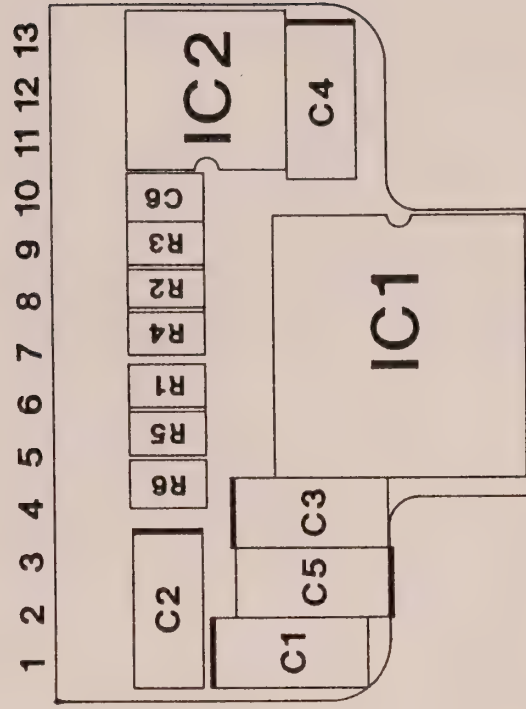


Figure 2 — Component Layout



---SET NEW DEFAULTS---

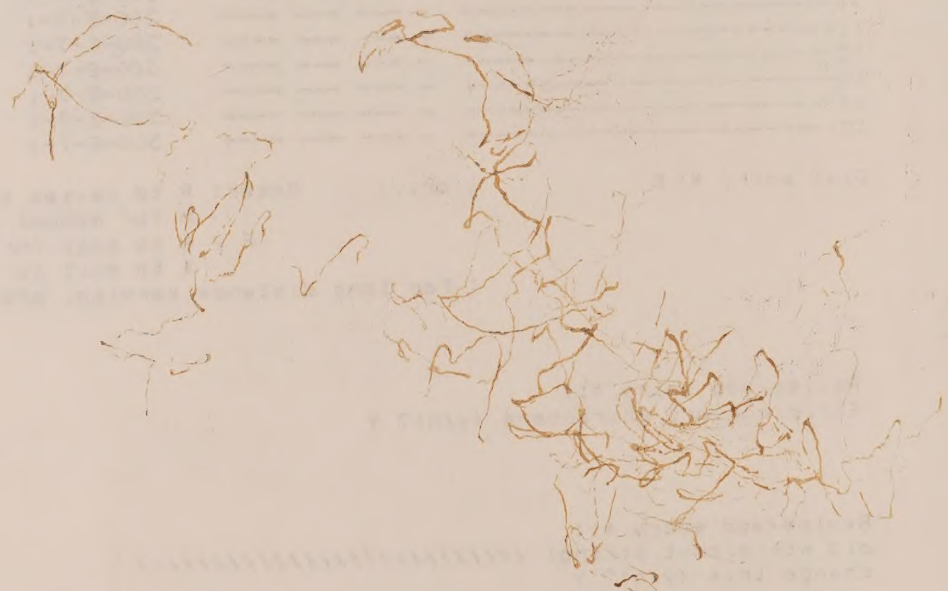
Present program defaults:

| | | | |
|--------------|------|-----------------|----------------|
| Baud rate | 1200 | SCREENDUMP FILE | A:SCRNDUMP.PCT |
| Parity | E | REDIAL DELAY | 20 |
| Data bits | 7 | CONNECT PROMPT | CONNECT |
| Stop bits | 1 | Line 25 help | Y |
| Echo | N | Foreground | 7 |
| Messages | N | Background | 0 |
| STRIP #1 | 019 | High inten. | 15 |
| REPLACE #1 | 0 | Print port | LPT1: |
| STRIP #2 | 0 | Print init. | '' |
| REPLACE #2 | 0 | Print width | 80 |
| STRIP #3 | 0 | Comm. port | COM1: |
| REPLACE #3 | 0 | Comm. init. | ,CS,DS |
| FACING P= | 0 | Modem init. | '' |
| LOGGED DRIVE | B: | C/R subst. |) |
| MARGIN WIDTH | 70 | | |

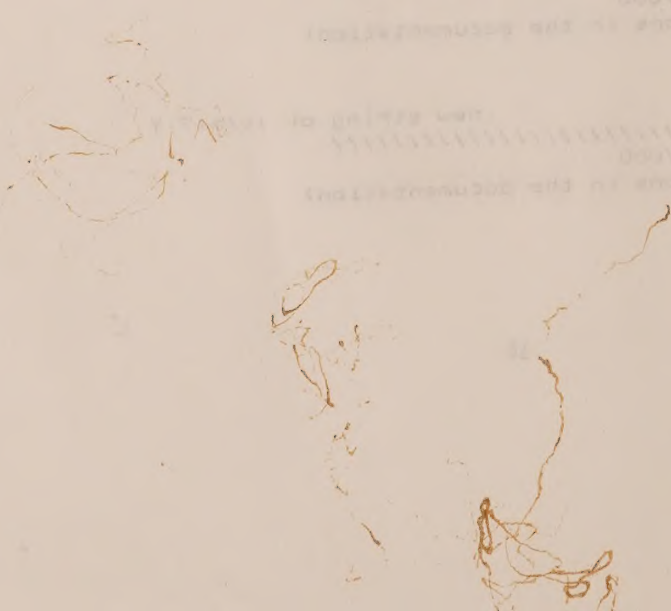
Enter DY to leave unchanged - <space>DY for 'null' value - <ESC>DY to quit
*** Enter new values



| RECEIVED | | | |
|------------|-------|-------------------|-------------|
| DATE | TIME | BY | REMARKS |
| 1944-12-15 | 10:30 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 11:00 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 11:30 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 12:00 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 12:30 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 13:00 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 13:30 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 14:00 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 14:30 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 15:00 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 15:30 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 16:00 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 16:30 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 17:00 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 17:30 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 18:00 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 18:30 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 19:00 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 19:30 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 20:00 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 20:30 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 21:00 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 21:30 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 22:00 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 22:30 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 23:00 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 23:30 | J. H. [illegible] | [illegible] |
| 1944-12-15 | 00:00 | J. H. [illegible] | [illegible] |



RECEIVED
DATE 1944-12-15
TIME 10:30
BY J. H. [illegible]
REMARKS [illegible]





**AVERY
DENNISON**

Office Products M
Brea, California 92821

Heavy Duty - 1"
EZD™ View Binder

| | |
|--------------|--------------|
| White #79789 | Gray #79409 |
| Black #79809 | Asst. #79410 |
| Heavy #79809 | |

